uc3m | Universidad Carlos III de Madrid

Escuela Politécnica Superior

Departamento de Ingeniería Mecánica

GRADO EN INGENIERÍA MECÁNICA

TRABAJO DE FIN DE GRADO

APLICACIÓN INFORMÁTICA SOBRE ANDROID PARA EL DISEÑO DE VOLANTES DE INERCIA

Autor: **Ulises Martín Díaz**

Tutor: **Higinio Rubio Alonso**

Leganés, 2017

Título: Aplicación informática sobre Android para el diseño de volantes de inercia.

Autor: Ulises Martín Díaz

Director: Higinio Rubio Alonso

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día…… de…….. de…….. en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

AGRADECIMIENTOS

Siendo este el último paso que debo dar para acabar el grado, miro atrás y veo todos los buenos recuerdos que durante estos cuatro años me ha dado la universidad. No todo ha sido un camino de rosas, pero el logro de acabar bien merece el esfuerzo y dedicación puesto.

Quisiera agradecer el apoyo recibido por parte de mi familia, si la cual no sería quien soy hoy.

Agradecer en especial a Higinio, por darme la oportunidad de desarrollar este proyecto con él y por todo el tiempo y dedicación que me dedicó.

RESUMEN

El presente proyecto consiste en el diseño de una aplicación para dispositivos multiplataforma con sistema operativo Android que ayude a una mejor y más fácil resolución de problemas de volantes de inercia.

Para la obtención del objetivo principal se ha creado un modelo capaz de calcular ciertos parámetros del diseño de volantes de inercia, analizándolos para determinar los resultados más relevantes. A fin de lograrlo, se empleará el entorno de programación para Android, "Android Studio", que permitirá crear una aplicación multiplataforma amigable, simple en ejecución pero a su vez compleja en cálculo ofreciendo así una precisa resolución del problema.

Con el fin de verificar los cálculos que la aplicación generará, se han testado varios problemas de diversas formas para un mayor sentido de la precisión. Entre los métodos escogidos se han usado: el método tradicional de hacerlo a mano, ejercicios comprobados en libros y programas de cálculo de volantes de inercia entre otros.

Para estar seguros de estar fuera de cualquier error en la ejecución de la aplicación, se han llevado a cabo ensayos en diferentes plataformas, tanto smartphones como tabletas; ya fuesen modelos físicos como simulados en ordenador.

ABSTRACT

This project consists of the design of an application for multiplatform devices with Android operating system that helps with a better and an easier resolution of flywheel problems.

To achieve the main objective, a model has been developed to calculate certain parameters of the flywheel design, analysing them to determine the key factors that define it. In order to do this the Android programming environment "Android Studio will be used. It will allow us to create a cross-platform friendly-user application, simple in execution but complex in computation, thus offering a precise resolution of the problem.

In order to verify the calculations that the application will generate, several problems have been tested in different forms for a greater sense of precision. Among the chosen methods it can be found: the traditional method of doing it by hand, checked exercises in books and former programs of calculation of flywheels among others.

To be sure of being out of any error while running the application, tests have been carried out on different platforms, both smartphones and tablets. Whether they were physical models or computer simulated.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1 Ámbito y motivación

Un volante de inercia es, por definición; un sistema de almacenamiento de energía mecánica, más específicamente; de energía cinética. La característica principal de un volante de inercia es la capacidad de recibir y dar energía en poco tiempo; ideal sistemas mecánicos de ciclos de energía discontinuos donde el periodo de tiempo sea muy corto. Su uso más convencional se encuentra en motores y compresores alternativos, prensas y troqueladoras, etc [8].

El concepto de almacenar energía mediante un volante de inercia data del 2400 aC. En el periodo neolítico, los egipcios los utilizaron para artesanías de alfarería y las espiras de los husillos. De hecho, este sistema de almacenamiento de energía fue ampliamente utilizado en la vida cotidiana, tanto en carros de guerra, bombas de agua e incluso generadores de energía.

Figura 1: Torno de alfarero.

Sin embargo, estos primeros volantes convencionales no eran tan eficientes en comparación con otros dispositivos de almacenamiento de energía dado que la gran cantidad de masa que tenían era muy costosa de mover para la cantidad relativamente escasa de energía almacenada, sin mencionar la corta capacidad de suministrar energía en el tiempo.

No es hasta la primera revolución industrial cuando los avances en los volantes fueron significativos. El paso que hubo en la construcción y en la fabricación de maquinaria en hierro fundido supuso para el volante de inercia el empujón que le faltaba, logrando así un mayor momento de inercia de masa y, por lo tanto, un ahorro significativo de su peso; lo que desembocó en la incorporación de los volantes en las máquinas de vapor.

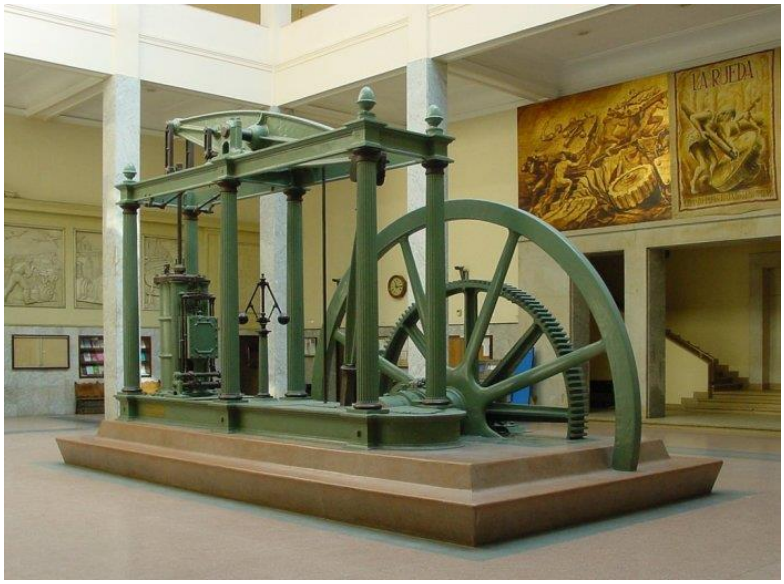

Figura 2: Máquina de vapor tipo Watt.

Aunque no es hasta las últimas décadas que presenciamos el alto rendimiento de los volantes que se desarrolló significativamente con una mejora notable como sistemas de almacenamiento de energía en una amplia gama de aplicaciones. Esto viene dado por el brutal avance de la tecnología y la informática, y como no; de la ciencia de los materiales.

A día de hoy, el impulso que se le da a la búsqueda y mejora de las tecnologías es, sin duda alguna; la mayor que se haya podido ver en la historia. Todo es tecnología, ya sea informática, de telecomunicaciones, electrónica, etc. El mundo de la ingeniería ha avanzado tanto en estos últimos años que lo que hoy es pinero mañana será obsoleto.

La implementación de potentes softwares de cálculo y diseño (CAD-CAM), ha permitido ser mucho más eficientes a la hora de gestionar recursos, más aun; al estar todo conectado, la mejora en todos los campos es pareja.

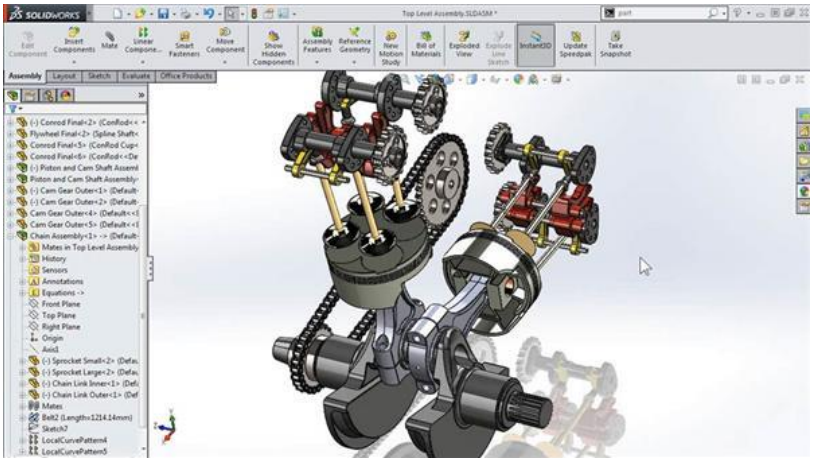

Figura 3: Pieza en el programa SolidWorks.

Uno de los ejemplos más conocidos y cotidianos es el de la telefonía móvil. Quien diría que en 1973 Martin Cooper, directivo de Motorola; realizaría la primera llamada desde un teléfono móvil, y que ahora, casi 44 años después, no haya un solo segundo en el que miles de personas en todo el mundo estén haciendo llamadas sin parar.

El sector de las comunicaciones y telefonía tiene su boom con los Smartphone. Un aparato en miniatura que actúa como un ordenador de bolsillo, con acceso a internet de múltiples formas, con capacidad de almacenar, generar, leer, enviar todo tipo de dato; incluso en ámbitos fuera de los de la comunicación.

Para hacerse una idea de lo que puede llegar a mover este gigante de la industria, en la Tierra hay alrededor de 7.000 millones de habitantes; se estima que a mediados del 2017, 5.000 millones tendrán un teléfono móvil. Siendo esto más del 71% de la población mundial total [21].

Dentro de este mega mercado, son varios los que se disputan la tarta. La competitividad que existe es descomunal y fiera; siendo cada porcentaje ganado o perdido una cuantiosa cantidad de dinero. En lo que a sistemas operativos refiere, según datos del tercer trimestre de 2015 la cuota de mercado se repartiría [20]:

- Android 72,6%
- IOS 20,2%
- Windows 6,4%
- Otros 0,8%

Como se puede observar de los datos, el sistema operativo Android es el más usado por los compradores, llegando a casi tres cuartas partes del total. Esto no solo ocurre en los teléfonos móviles, también pasa algo parejo con las tabletas, donde se pueden apreciar leves diferencias de porcentajes con respecto a los móviles. Pese a llevar ya dos años en decadencia, la industria de las tabletas sigue vendiendo un promedio de 175 millones de ejemplares anuales [19].

Los móviles son aparatos maravillosos, con infinidad de usos, no obstante; lo que de verdad hace útil y atractivo al móvil son las aplicaciones que puede albergar y usar. Una aplicación, o app, es un programa que se instala en el aparato para satisfacer una necesidad, ya sea de entretenimiento o no.

Figura 4: Aplicaciones creando la palabra “Apps”.

A día de hoy, las aplicaciones de contactos y redes sociales son las más usadas, teniendo millones de usuarios en todo el mundo. Seguido de ellas se encuentran las de juegos y entretenimiento. Además de las ya mencionadas hay aplicaciones para resolver problemas, tanto cotidianos, como podría ser un despertador, cámara de fotos, retoque de imágenes; como más técnicas, como una calculadora científica, diseño de sistemas de todo tipo, resolución de problemas, etc.

Se estima que la envergadura que adquirirá la industria de las aplicaciones en 2017 será de unas 270.000 millones de descargas de aplicaciones a nivel mundial, dejando una cifra de 150.000 millones de dólares [18].

Ateniéndose a todo lo mencionado previamente y viendo como la importancia e influencia de las nuevas tecnologías ejercen sobre el día a día, nace la principal motivación de este proyecto, al intentar integrar la mecánica de los volantes de inercia con la creación de una aplicación móvil. Este proyecto mecánico-informático, se fundamenta en el diseño e implementación de una aplicación móvil para sistema Android que ayude de una forma simple pero eficaz la resolución y cálculo de los volantes de inercia.

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Grado queda perfectamente planteado en el título del mismo como el desarrollo de una *"Aplicación informática sobre Android para el diseño de volantes de inercia"*. Se pretende que la aplicación informática sea multiplataforma, soportada por los principales smartphones y tablets del mercado que operan con Android.

La consecución de este proyecto engloba unos objetivos intermedios indispensables para lograr el objetivo principal, los cuales se definen a continuación:

- Recopilación actualizada de información relativa al tema tecnológico a desarrollar de volantes de inercia.
- Indagación en la existencia de aplicaciones o herramientas informáticas previas, similares a la que se realizará en este proyecto.
- Compendio de información de cómo desarrollar y crear aplicaciones sobre sistemas Android.
- Selección de un software genérico en el que implementar la aplicación.
- Diseño y desarrollo de una aplicación de prueba, para ver las posibilidades del software seleccionado.
- Creación de un modelo virtual capaz de interpretar y calcular sistemas de volantes de inercia, a partir de entrada de datos.
- Diseño y desarrollo de una aplicación multiplataforma con las siguientes características básicas: *nivel técnico de un ingeniero, intuitiva y de fácil uso para el usuario y de elevada funcionalidad*.
- Verificación legal: Comprobar si la aplicación informática creada cumple con la normativa vigente sobre las "app".
- Refinamiento de la funcionalidad, la estética en el diseño y la visualización de la aplicación.
- Realización de un manual de uso de la aplicación de cálculo de volantes de inercia.
- Diseño e implementación de un juego de pruebas (para los diferentes casos posibles) que verifiquen la correcta operatividad de la aplicación creada e informen de las posibles utilidades de la misma.

En cuanto a los medios que se emplearán para la realización de este Trabajo Fin de Grado, serán una serie de programas informáticos que se detallan a continuación:

- El programa ***Android Studio*** se usará como el entorno de programación para el desarrollo de nuestra aplicación Android. Android Studio es un entorno de desarrollo abierto, con integración vía web multiplataforma que permite su uso sobre el lenguaje Java, ampliamente utilizado para el desarrollo apps y de fácil manejo para principiantes en el mundo de la creación de apps. Como ya se explicará más adelante, esta elección se hizo en base a su gran versatilidad y número de vías de solicitar ayuda.

- El software ***Snagit Editor*** se empleará para modificar imágenes para su posterior utilización en el proyecto.

- El programa de parametrización ***Solid Edge*** se usará en la creación de modelos 3D propios para imágenes para el manual de uso y ejemplos que se pongan.

- Por último, se empleará el conjunto de software ***Microsoft Office***: para la comprobación de los resultados de la aplicación se usará la herramienta de hoja de cálculos de ***Excel*** y para la redacción del informe se utilizará el editor de textos ***Word***.

1.3 Etapas del proyecto

Todo proyecto debe estar regido por unos cánones, unas directrices las cuales seguir para llegar a buen puerto. Este trabajo de fin de grado, al ser innovador; los pasos a seguir no necesitan ser muy rígidos, ya que se admite una cierta flexibilidad. Desde un primer paso donde se piensa la apariencia y primeras funciones, hasta una puesta en marcha y definición más rigurosa, analizando datos y comprobando que se alcanzan los objetivos fijados.

Teniendo todo en cuenta se podrían discernir tres etapas:

1) Recolección de información y ensayos. En la primera etapa se fijarán los objetivos, motivaciones y necesidades que la app debe cumplir. Se realizará un boceto y se analizará su funcionalidad y viabilidad.

2) Diseño y análisis. Tras ver que ese primer dibujo conceptual es válido, se pasará a un diseño más detallado, profundizando en las funciones que se utilizarán y de cómo se llegarán a los resultados.

3) Implementación final. Para concluir se realizarán comprobaciones con distintos métodos para verificar finamente que el producto creado está en las expectativas de los objetivos fijados inicialmente.

1.4 Estructura del documento

El planteamiento de este Trabajo de Fin de Grado queda establecido en ocho apartados, incluyendo las *"Referencias"* como una de las divisiones.

- **Introducción.**

  Aquí se hablará del ámbito y motivaciones que se han tenido para el desarrollo de la aplicación, se fijarán los objetivos del proyecto y se comentarán las diferentes divisiones de las que consta la estructura de la memoria.

- **Teoría de volantes de inercia.**

  En este apartado se explicará toda la teoría relacionada con los volantes de inercia, detallando como se irían calculando los parámetros más significativos.

- **Descripción de las herramientas software empleadas.**

  Cada una de las herramientas, en este caso programas informáticos; utilizadas en el desarrollo del proyecto será explicada para un mejor entendimiento de su utilización.

- **Metodología.**

  Dentro de este apartado se encontrarán los antecedentes a este proyecto, la normativa aplicable a las aplicaciones, el planning y tiempos y los resultados esperados del mismo.

- **Desarrollo de la aplicación.**

  En este capítulo, se profundizará en los pasos dados para llegar a la app final y de cómo se ha utilizado Android Studio para lograrlo.

- **Presupuesto y entorno socioeconómico.**

  Una breve descripción de los gastos que ha conllevado la creación de la aplicación, junto con una referencia al estado socioeconómico del mundo de las aplicaciones y la telefonía móvil.

- **Conclusiones y trabajos futuros.**

  En la penúltima etapa se comprobarán que se han cumplido los objetivos, y se enumerarán las mejoras que se le puedan aplicar a la app.

- **Referencias.**

Por último, se pondrán los sitios y papeles consultados para el desarrollo de este Trabajo de Fin de Grado.

- **Anexos.**

En este apartado, no contabilizado en las ocho divisiones; se ilustrarán algunos procesos que no hayan sido profundizados tanto a lo largo de la disertación con imágenes.

2. TEORÍA DE VOLANTES DE INERCIA

2.1 Introducción

Un volante de inercia es un mecanismo consistente en una masa adicional, cuya forma se asemeja a la de una rueda. Su momento de inercia es tal que puede ser sustituido por una masa puntual localizada en el punto donde se sitúa la masa reducida de la máquina. La energía cinética que alberga el punto de la masa reducida es el mismo que la del conjunto de todos los miembros, al igual que se reduce al mismo punto las fuerzas aplicadas [17][15].

Mediante un movimiento rotativo, el volante acumula energía cinética para reducir las oscilaciones en el par torsor que actúa sobre el eje en el que va montado a lo largo de su ciclo. Las variaciones de la velocidad de giro producidas por las oscilaciones pueden influir en el buen funcionamiento de la máquina, por lo que el uso de los volantes ayuda a su atenuación.

Figura 5: Volante de inercia.

Al no ser muy frecuente tener un régimen cíclico uniforme en una máquina, mediante el uso de volantes se hace que las velocidades extremas (ω_{max} y ω_{min}) a lo largo del ciclo, no se desvíen mucho con respecto de la velocidad media (ω_m) aumentando así la inercia de la máquina.

Los objetivos principales del volante son, disminuir la amplitud de fluctuación de la velocidad, reducir la amplitud del par torsor fluctuante y aportar la energía a un ritmo lento y constante o almacenarla cuando la energía esté disponible con el objetivo de volverla a suministrar cuando se requiera [15].

Entre los usos más frecuentes del volante de inercia se encuentran:

- En las frenadas de los vehículos, absorbiendo y almacenando la energía de manera que luego se pueda usar en la aceleración del mismo (KERS).
- La superación de los puntos muertos de los motores de los vehículos, ya que de las dos vueltas del ciclo, solo en una se genera par.
- Como dispositivos para atenuar el funcionamiento de plantas generadoras de electricidad tales como la eólica o la fotovoltaica, al igual que para otras utilidades industriales.
- En prensas, punzadoras u otras máquinas donde se requiere de una solicitación muy elevada en un breve periodo de tiempo.

2.2 Terminología de los volantes de inercia

Como se muestra en la figura siguiente figura, los volantes de inercia convencionales constan de las siguientes partes:

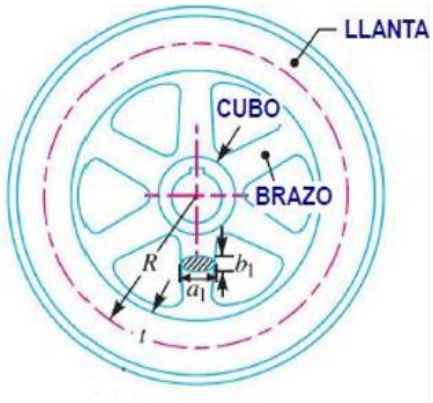

Figura 6: Vista frontal de un volante de inercia.

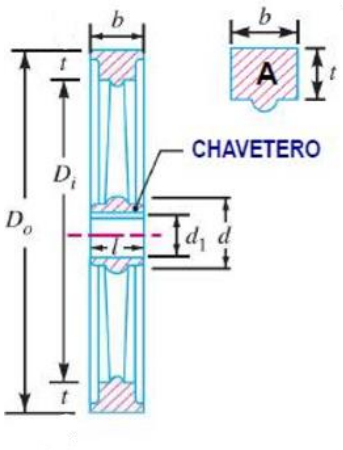

Figura 7: Vista de canto de un volante de inercia.

Dependiendo del uso que se le vaya a dar y del régimen de rotación, el número de brazos y la forma y número de agujeros variará. No obstante, todo será simétrico ya que una diferencia, por muy leve que sea; en el reparto de masas generaría unas fuerzas tales que terminarían rompiendo los apoyos donde estuviera el eje del volante. Esta es la razón por la que los volantes de inercia no simétricos no se desarrollaron, quedándose solo en la fase experimental.

Dependiendo de sus dimensiones y uso se podrían dividir en tres categorías por su tamaño [13] y en dos por su régimen de velocidad [8] (se hace referencia a los volantes de inercia convencionales).

Tamaño	Dimensiones	Nº Piezas	
Grandes	> 2500 mm	2	Por su gran tamaño y costosa fabricación, se divide en dos partes.
Medios	600- 2500 mm	3	Las tres partes son: llantas cubo y brazos.
Pequeños	< 600 mm	1	Hechos de una sola pieza, se le añaden los agujeros de manera simétrica.

Tabla 1: Tipología de los volantes por tamaño.

Tipo	Complejidad	
Baja velocidad	Poca	Su diseño se basa en cilindros macizos de tal forma de aumentar la masa y así aumentar la energía almacenada.
Alta velocidad	Alta	El objetivo es maximizar la velocidad (rpm) sin que el material se desgaste debido a las altas fuerzas centrífugas. Por lo que elevar lo máximo posible el coeficiente de la tensión radial obliga a utilizar materiales compuestos.

Tabla 2: Tipología de los volantes por velocidad.

2.3 Tipos de volante de inercia y su uso

Un volante de inercia de una masa por si solo es bastante sencillo, ya que no consta de partes complejas ni de un gran número de piezas, no obstante no es el único tipo que se puede encontrar en el mercado hoy en día. A menudo se suelen encontrar con dispositivos montados sobre él, un ejemplo muy extendido sería el que tienen los coches.

Todos los motores de los coches incorporan un volante de inercia para poder superar la vuelta en la que no generan par, ya que esa vuelta “muerta” no hace más que frenar las revoluciones del motor, provocando una modificación de la velocidad de rotación. Al volante que se puede apreciar en la figura, se le ha acoplado a la barra de transmisión que alberga a los pistones; de esta manera, cuando haya fases de energía podrá absorberla y cuando la fase no produzca podrá devolverla para no hacer variar la rotación del eje.

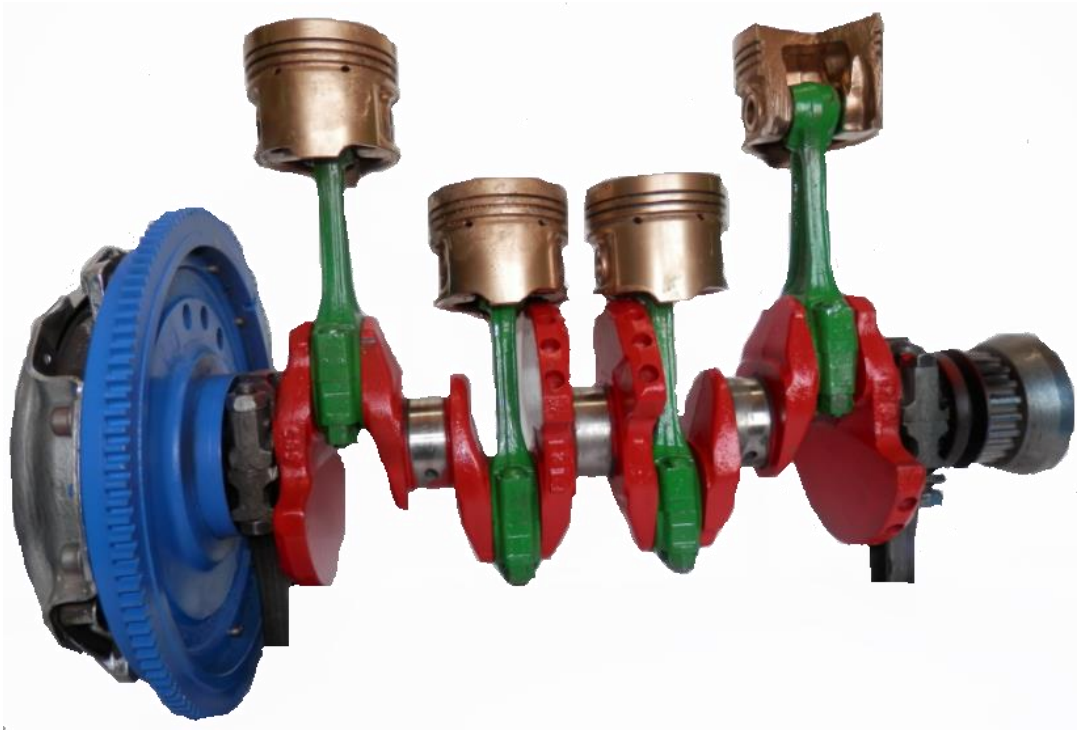

Figura 8: Volante de inercia de un automóvil acoplado a un eje de transmisión.

Otros modelos más complejos y sofisticados incorporan más de una masa para poder atenuar aún más las oscilaciones generadas por la máquina. El volante bimasa es un volante de inercia con dos masas (la masa principal con la corona dentada para el arranque y la secundaria para el lado de la transmisión con las ranuras de ventilación para la evacuación del calor del embrague), este en comparación con el de una única masa, filtra las oscilaciones torsionales con sus sistemas de resortes y amortiguación y las absorbe casi por completo.

2.4 Fuerzas que intervienen en los mecanismos

Toda máquina, por el hecho de tener un objetivo mecánico, es decir; hacer un movimiento concreto, generará una fuerza o par a raíz de ese movimiento, haciendo así que en otra parte de la máquina se sufra/genere otra fuerza o par. Las fuerzas y pares se pueden clasificar en tres grupos dependiendo de su origen [17][15]:

1.- Fuerzas y pares internos. Estas fuerzas se generan por el efecto de los eslabones, no obstante; al considerarlos sólidos rígidos indeformables, además de estar en equilibrio; en el balance de energías no aparecerán ya que no generan trabajo.

2.- Fuerzas y pares externos. Las fuerzas exteriores que se aplican directamente sobre distintos eslabones del mecanismo. Se pueden dividir en:

- Peso de los eslabones. Es habitual no contar con este tipo de solicitación para el cálculo total dado que los eslabones se diseñan con la menor masa posible, puesto que estarán sometidos a altas revoluciones, generando así elevadas fuerzas e inercias indeseadas. En caso de estar dimensionados con medidas importantes, sí habría que contar con su aportación.

- Fuerzas y pares motores. Son los responsables del movimiento y realizan un trabajo positivo. Este tipo de fuerza es por la que se diseña la máquina.

- Fuerzas y pares resistentes. Se oponen al movimiento y realizan un trabajo negativo en el ciclo. Si son causados por la fricción se consideran fuerzas pasivas y, si se pueden aprovechar para la realización del trabajo se llaman fuerzas y pares útiles.

3.- Fuerzas y pares de inercia. Son aquellas generadas por las aceleraciones. Cuando se aceleran, el trabajo que dejan es positivo y cuando se deceleran el trabajo es negativo. Al finalizar el ciclo, la suma de ambas partes es nula, por lo que no consumen ni aportan energía.

2.5 Masa reducida y teorema de las fuerzas vivas

Una masa reducida es un concepto por el cual la resolución de problemas mecánicos integrados por múltiples masas se simplifica a una sola. Esta masa se sitúa en el punto conveniente en el que se reduzcan todas las fuerzas y por consiguiente posea la misma energía que el conjunto del resto de miembros [17][15].

Una vez se tiene la masa reducida se puede aplicar el teorema de las fuerzas vivas. Dicho teorema enuncia que el trabajo generado por las fuerzas vivas que actúan sobre un mismo conjunto, es igual a su energía cinética. El sumatorio de las fuerzas del conjunto se denomina fuerza resultante o neta.

En este caso aplicado a un volante de inercia, el punto de masa reducida (A) se situará en la periferia del volante, quedando solo un sistema al cual se aplica un esfuerzo motor y otro resistente.

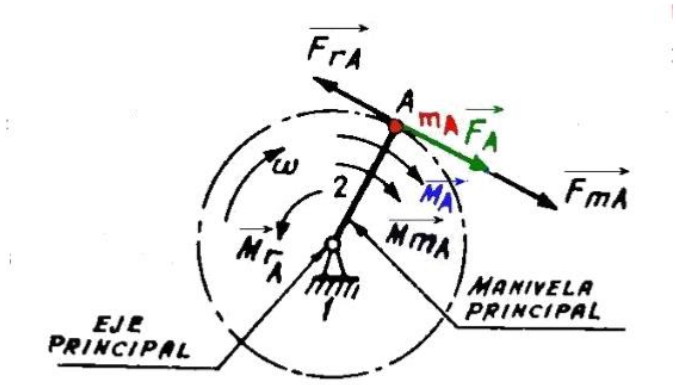

Figura 9: Equivalente dinámico de 1GDL [17].

m_A= masa reducida al punto A

$\vec{M}_A$= Par reducido en A

$\vec{F}_A$= Fuerza reducida en A

$\vec{M}_{rA}$= Par resistente reducido en A

$\vec{M}_{mA}$= Par motriz reducido en A

$\vec{F}_{rA}$= Fuerza resistente reducida en A

$\vec{F}_{mA}$ = Fuerza motriz reducida en A

r=distancia del punto de reducción al eje

ω = velocidad angular del eje

Si se hacen los balances de fuerzas y pares reducidos se obtendría:

$$\vec{F}_A = \vec{F}_{mA} - \vec{F}_{rA}$$

$$\vec{M}_A = \vec{M}_{mA} - \vec{M}_{rA}$$

Los diagramas de un mecanismo con estas similitudes se podría expresar:

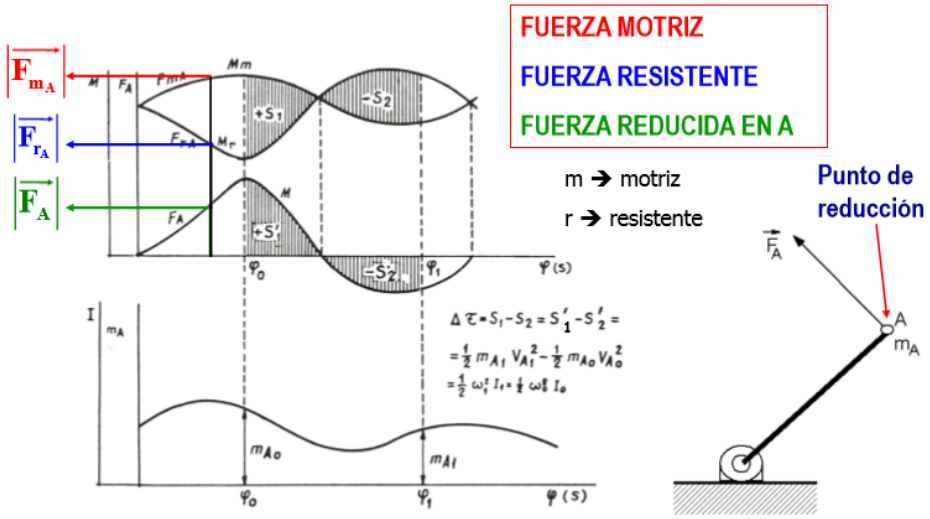

Figura 10: Diagramas de fuerza.

Dado un mecanismo, la masa reducida y el momento de inercia reducido al eje principal son independientes de la velocidad adquirida:

$$I = \frac{V_A^2 \, M_A}{\omega^2} = M_A \, r^2$$

En cada posición de la máquina, la energía cinética valdrá:

$$E = \frac{1}{2} \, m_A \, v_A^2 = \frac{1}{2} \, I \, \omega^2$$

Al ir de la posición 1 a la 2, la variación de energía cinética será igual al trabajo de la fuerza neta a lo largo del mismo periodo de tiempo.

2.6 Etapas de marcha de una maquina cíclica

Se dice de una máquina cíclica cuando sus diagramas de posición respecto al tiempo están formados por bloques que se repiten, estos bloques se denominan ciclos cinemáticos. En la mayoría de las máquinas, los ciclos cinemáticos coinciden con los ciclos de trabajo. No obstante en algunas se da el caso donde un mismo ciclo de trabajo influye en varios ciclos cinemáticos. Se pueden establecer tres periodos de funcionamiento [17]:

1.- Arranque. Al partir del reposo, la velocidad inicial es $\omega_1 = 0$. Al transcurrir un tiempo, ω_2 valdrá ω y obtendremos:

$$T_m = T_r + \frac{1}{2} I \omega^2$$

Donde,

T_m es el trabajo motor

T_r es el trabajo resistente

I es la inercia

ω es la velocidad angular

El trabajo motor es la suma del trabajo resistente realizado más el trabajo que se almacena en la máquina al incrementar la velocidad de sus eslabones. A este trabajo se le llama trabajo de inercia. Durante el periodo de arranque se verificará que: $T_m > T_r$.

2.- Régimen permanente. La velocidad se mantiene constante ($\omega_1 = \omega_2 = \omega$). Obteniendo:

$$\frac{1}{2} I \omega_1^2 - \frac{1}{2} I \omega_2^2 = 0$$

Por lo tanto en ese periodo de tiempo $T_m = T_r$

3.- Parada. La velocidad final $\omega_2 = 0$. Por lo que:

$$T_r = T_m - \frac{1}{2} I \omega^2$$

En este último período se devolverá todo el trabajo acumulado durante la puesta en marcha, el cual se almacenó en forma de energía cinética. En este período $T_m > T_r$.

2.7 Variaciones cíclicas de la velocidad

Para un instante concreto, la potencia desarrollada por la máquina es [17][15]:

$$W = M\,\omega = F_A\ v_A$$

Donde,

M= par total reducido ($M=F_A\ r$)

El par total reducido no se mantiene constante durante el ciclo de la máquina, este variará de acuerdo con las diferentes modalidades de cada una. Como consecuencia, el trabajo desarrollado en las distintas etapas del ciclo tampoco será constante y la energía cinética será distinta para cada posición de la máquina. Como consecuencia, la velocidad angular ω fluctuará y se producirá una aceleración. Con todo ello, en cada posición de la máquina se tendrá un estado dinámico distinto. Concluyendo así que el movimiento de una máquina en cada parte del ciclo no será uniforme.

2.8 Aplicación del teorema de las fuerzas vivas

Como ya se mencionó previamente, el teorema de las fuerzas vivas nos dice que el trabajo realizado en un intervalo de movimiento es igual a su variación de energía cinética [17][15][5].

El trabajo T_{12} realizado por las fuerzas exteriores entre los instantes 1 y 2 del movimiento, será:

$$T_{12} = E_{c2} - E_{c1}$$

Donde,

E_{c1} es la energía cinética en el instante1

E_{c2} es la energía cinética en el instante2

Como ya se mencionó en el apartado de *"Fuerzas que intervienen en los mecanismos"*, el trabajo T durante el intervalo 1-2 consta de:

- Trabajo motor (T_m): Producido por las fuerzas y pares motrices aplicadas al mecanismo para producir el movimiento.

- Trabajo resistente (T_r): Provocados por fuerzas y pares que se oponen al movimiento. Se distinguen dos tipos:

  - Trabajo útil (T_u): Es el producido por las fuerzas que hay que vencer para realizar el movimiento para el cual se diseñó el mecanismo.

  - Trabajo pasivo (T_p): Es el que se realiza para vencer la fricción.

$$T_{12} = T_m - T_r = T_m - T_u - T_p$$

La energía cinética se obtiene haciendo el sumatorio de la energía producida por las masas en movimiento, tanto de traslación como de rotación.

$$E_c = \frac{1}{2} \sum m\, v^2 + \frac{1}{2} \sum I\, \omega^2$$

Utilizando el momento de la máquina reducido a un eje, el cual tiene la misma energía cinética que el mecanismo completo, se puede simplificar la expresión para obtener:

$$E_c = \frac{1}{2} I\omega_2^2 - \frac{1}{2} I\omega_1^2$$

Para un desplazamiento angular infinitamente pequeño, tenemos:

$$M\ d\varphi = \frac{1}{2}\ d(I\ \omega^2)$$

Donde,

$d\varphi$ es el diferencial de la posición de la máquina.

Como tanto I como M son variables dependientes de la posición φ de la máquina, el momento M será:

$$M = I\ \varepsilon + \frac{1}{2}\ \omega^2\ \frac{dI}{dt}$$

Donde,

ε es la aceleración angular

El segundo término de la expresión representa como influye la distribución de las masas en la inercia de la máquina. En la mayoría de los casos, el primer término es bastante menor que el segundo, pudiendo así simplificar la expresión de la siguiente manera:

$$M = I\ \varepsilon = M_m - M_r$$

Se puede concluir que, en los instantes en el que el momento M se anule, la aceleración angular ε tendrá que ser igual a cero, independientemente del valor del momento de inercia. Si el movimiento es uniforme ε valdrá cero. Por lo tanto, quedará que el par motor será igual que el par resistente.

$$M_m - M_r = 0$$

2.9 Régimen permanente

Se dice que una máquina opera bajo régimen permanente cuando en cada fase del movimiento se corresponde un estado determinado de la máquina, es decir; está en régimen permanente si tiene la misma velocidad media en todos los ciclos (I_ϕ=I_o y ω_ϕ= ω_o) [17][15].

El valor medio de la velocidad se denomina velocidad de régimen y viene expresado mediante la siguiente fórmula:

$$\omega = \frac{\int_0^{\Phi} \omega \, d\varphi}{\Phi}$$

Si ahora se aplica el teorema de las fuerzas vivas en el intervalo (0 - ϕ), se obtiene:

$$T_\Phi = \int_0^{\Phi} M \, d\varphi = E_\Phi - E_0 = \frac{1}{2} I_\Phi \omega_\Phi^2 - \frac{1}{2} I_0 \omega_0^2 = 0$$

Esto viene a representar la ecuación del régimen de permanencia en una máquina. En el conjunto del ciclo, el par motor total es igual al par resistente total. Con lo que la suma de las áreas encerradas por las curvas de M_m y M_r debe ser cero.

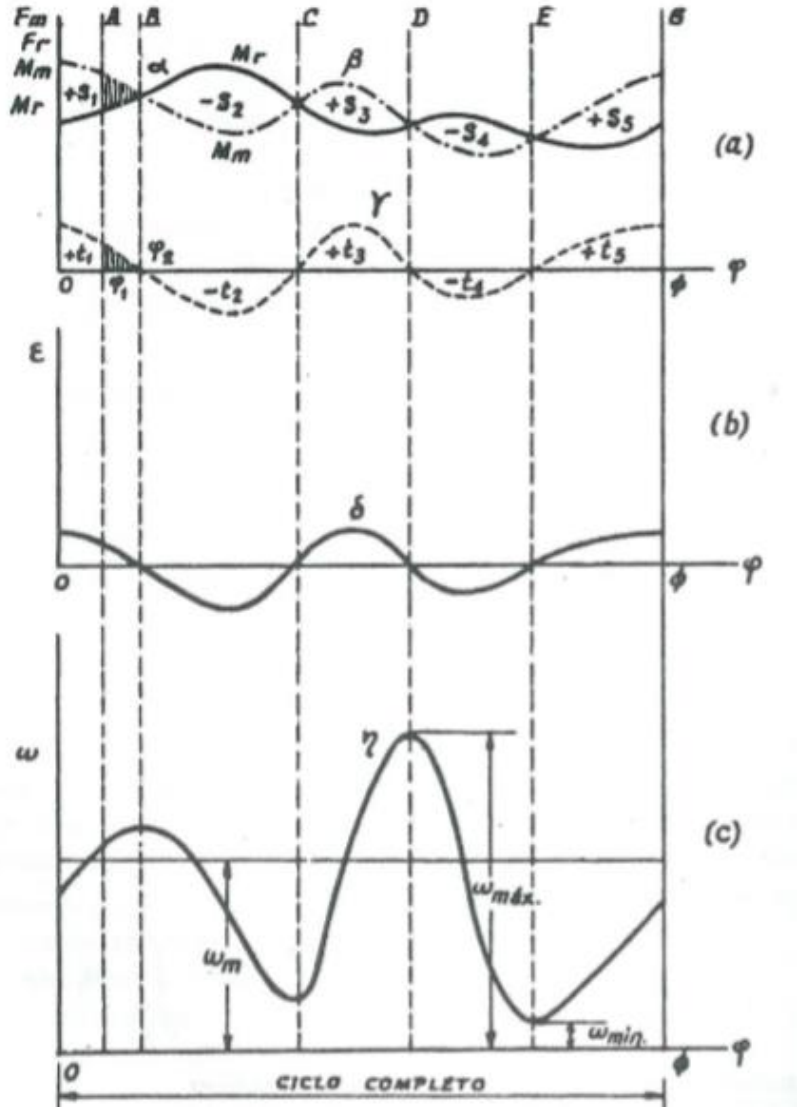

Figura 11: Diagramas de par de un ciclo entero.

Como se puede observar de la figura anterior, la suma de las áreas s_1, s_2, s_3, s_4, y s_5, es el trabajo motor del ciclo, y la suma de las áreas t_1, t_2, t_3, t_4, y t_5, el correspondiente al trabajo resistente del mismo, son iguales. Esto hace que la máquina no se salga de su régimen de permanencia e indica que la energía transmitida a la máquina en un ciclo es completamente consumida.

Cumpliéndose esto, la máquina empieza siempre cada ciclo con la misma energía cinética y con la misma velocidad, haciendo así que la velocidad media a lo largo de todos los ciclos permanezca constante.

2.10 Cálculo del grado de irregularidad

Como se ha comentado y demostrado anteriormente, una máquina en régimen permanente no sufre ningún incremento en el trabajo o en la energía cinética en cada uno de los sucesivos ciclos. No obstante, en los casos reales el par motor y el resistente no son iguales a lo largo del ciclo. Esto se ilustra mejor en la siguiente figura [17][15]:

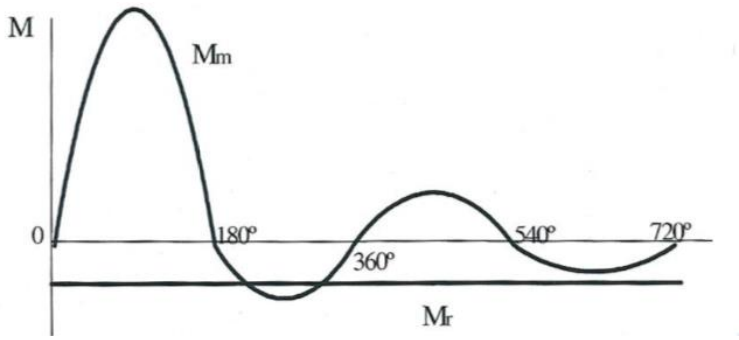

Figura 12: Ciclo de par real de una máquina.

Al producirse estas variaciones en los momentos, la velocidad también sufrirá cambios. Para el diseño de un volante de inercia es necesario fijar la máxima variación de velocidad permitida durante un ciclo. Para ello se define el grado de irregularidad o coeficiente de fluctuación como:

$$\delta = \frac{\omega_{max} - \omega_{min}}{\omega_m}$$

Donde,

ω_m es la velocidad angular media

$$\omega_m = \frac{\omega_{max} + \omega_{min}}{2}$$

El grado de irregularidad mide la desviación relativa que experimentará la velocidad de régimen, la cual indica el efecto que tiene el volante sobre la máquina.

Tipo de máquina	δ
Bombas alternativas	0,04
Bombas centrífugas	0,02
Prensas mecánicas	0,04
Máquinas de serrar	0,04
Hiladoras y telares	0,01
Máquinas herramienta	0,03
Maquinaria agrícola	0,05
Maquinaría de obra	0,05
Alternadores autónomos	0,004
Alternadores autónomos o a la red	0,001
Dinamos	0,005

Figura 13: Tabla de irregularidades de algunas máquinas.

Como se puede ver, la figura anterior muestra un rango de los valores que puede tener δ dependiendo del uso que tenga la máquina.

2.11 Cálculo aproximado de un volante de inercia

Para el cálculo aproximado del volante, se hace la hipótesis de considerar despreciable la inercia de la máquina respecto a la inercia del volante. Si se aplica esta hipótesis a la ecuación de las fuerzas vivas e integramos, obtenemos [5][17][15]:

$$A = \int_{\varphi_1}^{\varphi_2} M\, d\varphi = \frac{1}{2} I_V \omega_{max}^2 - \frac{1}{2} I_V \omega_{min}^2 = \frac{1}{2} I_V (\omega_{max} - \omega_{min})(\omega_{max} + \omega_{min})$$

Despejando la ecuación se obtiene:

$$A = \delta\, I_V\, \omega_m^2$$

$$I_V = \frac{\delta\ \omega_m^2}{A}$$

$$\delta = \frac{A}{I_V\ \omega_m^2}$$

Donde,

A es el trabajo entre ω_{max} y ω_{min}, siendo la suma algebraica de las áreas comprendidas entre las dos abscisas anteriores

De la ecuación sacamos que el grado de irregularidad disminuirá a medida que el área sea menor y la velocidad media aumente. Esto indica que cuando haya más de un eje donde colocar el volante en una máquina, se deberá colocar en el que gire a mayor velocidad.

Para calcular el volante se necesitará conocer el ciclo del trabajo (A), el grado de irregularidad (δ) y la velocidad de régimen (ω). Para este último parámetro, se hace la asunción de que la velocidad de régimen es la misma que la velocidad media; esto no es cierto en todos los casos, pero la diferencia que se encuentra entre ambas no es muy significativa.

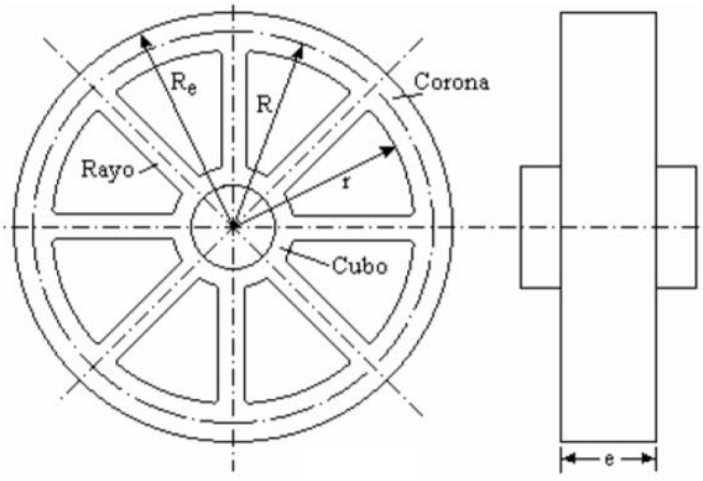

Figura 14: Boceto de un volante de inercia.

El valor que se proporciona una vez se diseña el volante es el factor de inercia (PD^2), siendo P el peso del volante y D el diámetro medio de la llanta del volante, donde se supone que se concentra toda la masa de la llanta.

$$I_V = m\ R^2 = \frac{P}{g}\frac{D^2}{4} = \frac{A}{\delta\ \omega_m^2}$$

$$P\ D^2 = 4\ g\ I_V = \frac{4\ g\ A}{\delta\ \omega_m^2}$$

Para hacer el cálculo del volante, se pueden conocer las dos abscisas para calcular A, pero se desconoce el valor de ω_{max} y ω_{min}, que sólo se podrán hallar una vez determinado el volante.

Para identificar cuáles son las abscisas correspondientes a ω_{max} y ω_{min}, utilizadas para calcular A, se observa cuál es la mayor área positiva (+S) y la mayor área negativa (-S), en valor absoluto.

2.12 Método aproximado rectificado de un volante de inercia

De la forma que se ha procedido anteriormente, la acción de las masas de la máquina se ha prescindido para el cálculo de la inercia, evadiendo así el término de la inercia de las masas. Esta inercia no es despreciable en algunos casos, pudiendo llegar a interferir en los resultados obtenidos, se debe rectificar el método anterior [15][17].

Si el momento de inercia reducido (I_R), no es despreciable frente al momento de inercia del volante (I_V), el momento de inercia del volante (I_V) se calculará restando al momento de inercia total (I) el momento reducido de la máquina.

$$I_V = I - I_R$$

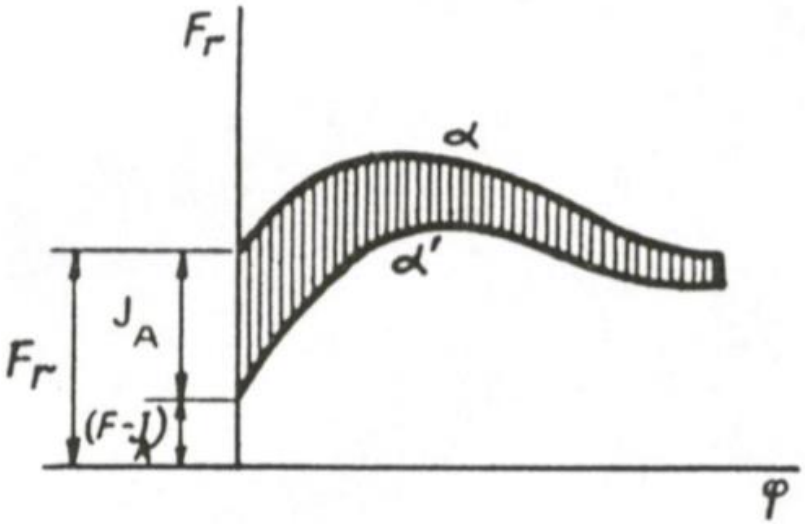

Figura 15: Curvas de esfuerzos resistentes.

De este modo, se rectifica el método anterior. Sea α la curva de esfuerzos resistentes reducidos, se calcula el valor del esfuerzo de inercia J_A de todas las posiciones de la máquina, reducidos al punto A. Si restamos este valor J_A a la curva α, obtenemos una nueva curva de esfuerzos resistentes α'. Y, finalmente, se procede del mismo modo que en el caso anterior.

El método aproximado, ya se haya hecho la rectificación o no, ofrece suficiente aproximación con máquinas que van a elevada velocidad. Sin embargo, para máquinas lentas, se obtienen volantes muy superiores a los que se necesitan en realidad.

2.13 Cálculo exacto del volante. Método de wittembauer

En el cálculo del volante de inercia, conocemos las variaciones de la energía cinética y del momento de inercia reducido de la máquina sin volante a lo largo de todo un ciclo. La energía cinética para cualquier posición, es [2][17][15]:

$$E_c = \frac{1}{2} I_R \omega_R^2$$

Por lo tanto, podemos ver que el incremento de energía se puede deber tanto a las variaciones del momento de inercia reducido como a las variaciones de velocidad angular. Como:

$$\Delta E_C = E_C - E_O \qquad I_C = I - I_V$$

Si se hace un cambio de escala, se pueden representar en la misma gráfica, haciendo uso del diagrama de Wittenbauer:

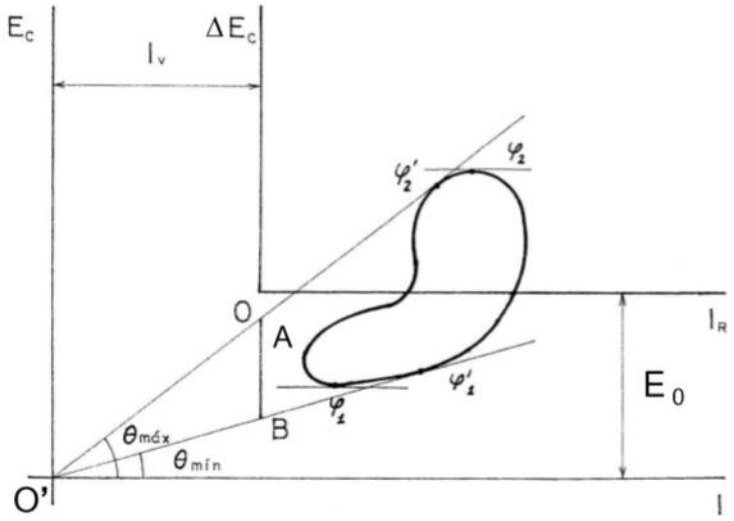

Figura 16: Diagrama de Wittenbauer.

$$E_C = \Delta E_C + E_O \qquad I = I_C + I_V$$

De este modo, si queremos calcular la velocidad de la máquina en una posición φ, se tiene que calcular la pendiente de la recta que une el origen O' con el punto de la gráfica que representa la posición φ. Haciendo esto, y conociendo el valor de θ, se puede sacar el valor de la velocidad angular:

$$tg\theta = \frac{E_c}{I} = \frac{\omega^2}{2} \qquad \omega = \sqrt{2\, tg\theta}$$

El punto O' se hallará mediante la intersección de las dos tangentes a la curva con estas pendientes:

$$tg\theta_{max} = \frac{\omega_{max}^2}{2} \qquad tg\theta_{min} = \frac{\omega_{min}^2}{2}$$

Medimos la distancia AB, correspondiente a la intersección de las dos pendientes con el eje de ordenadas.

$$\overline{AB} = I_V\, tg\theta_{max} - I_V\, tg\theta_{min} = I_V\,(tg\theta_{max} - tg\theta_{min})$$

$$= I_V \left(\frac{\omega_{max}^2}{2} - \frac{\omega_{min}^2}{2}\right) = \frac{I_V}{2}\,(\omega_{max} - \omega_{min})(\omega_{max} + \omega_{min})$$

$$= I_V\, \omega_m^2\, \delta$$

Finalmente se obtiene:

$$I_V = \frac{\overline{AB}}{\delta\, \omega_m^2}$$

2.14 Intervención el volante en la marcha de la máquina

Tal y como se ha ido explicando, el objetivo del volante es el de limitar la variación cíclica de la velocidad, manteniéndola dentro de unos límites acotados por el grado de irregularidad de la máquina [17][15].

Si por ejemplo tenemos un alternado de 1800 rpm que debe funcionar con un grado de irregularidad de 0.008, las velocidades de los extremos serán:

$$\omega_{max} = \omega_m\,(1 + \delta) = 1800\,(1 + 0{,}008) = 1814{,}4\,rpm$$

$$\omega_{min} = \omega_m\,(1 - \delta) = 1800\,(1 - 0{,}008) = 1785{,}6\,rpm$$

La velocidad se desviará de su valor medio como mucho un 0.8%. Implicando que la máquina funcionará prácticamente como si lo hiciera con velocidad constante de 1800 rpm.

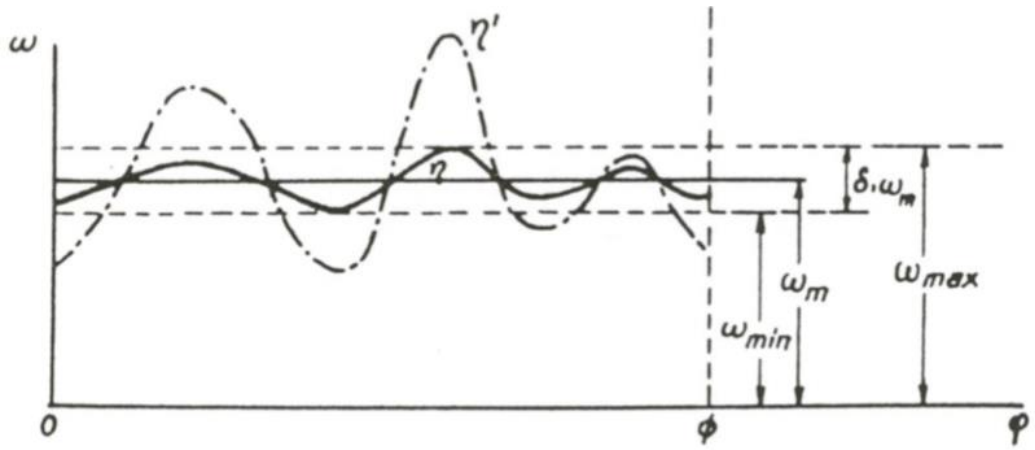

Figura 17: Diagrama del efecto de un volante acoplado a una máquina.

El funcionamiento de la máquina sin volante de inercia se asemejaría al de la curva η', mientras que la misma maquina con el volante incorporado sería más del estilo del de la curva η.

En los tramos donde el par motor es superior al par resistente, se dispondrá de un exceso de energía motriz. El cual se almacenará en el volante en forma de energía cinética. Por el contrario, en los momentos en el que el par resistente supere al par motor, el déficit de energía se compensa con la energía que el volante ha absorbido en el tramo anterior. El volante de inercia corrige los incrementos de energía dando y recibiendo a su debido momento, haciendo que el ciclo esté más controlado y sea más constante

3. DESCRIPCIÓN DE LAS HERRAMIENTAS SOFTWARE EMPLEADAS

3.1 Introducción

Estando en un mundo donde la cantidad de opciones a elegir es tan basta, la decisión de ir por un camino u otro es crucial. Dentro del tema a desarrollar en este proyecto, se plantea la siguiente duda: "¿Qué método utilizar para desarrollar la app?".

Esta ardua tarea viene influenciada por muchos factores, tales como el número de personas al que va ir destinado, cómo presentarlo y a qué nivel de complejidad, tiempo disponible para su implementación, conocimientos previos en el mundo de la informática…

Como se podrá apreciar en los siguientes apartados, la elección del programa final de desarrollo junto con el resto de cuestiones, se detallará de manera que se denote que la opción elegida es la más idónea, no solo para este proyecto sino también para la situación actual.

3.2 Sistema operativo

A día de hoy, existen distintos sistemas operativos, y cada uno requiere de un lenguaje y conocimientos específicos. Pese a poder crear la misma aplicación para los diferentes sistemas, dando los mismos resultados y opciones; no es extrapolable ni el código ni la manera de llevar a cabo su implementación. Para poder saber cuál es la mejor opción en cuanto al número de personas a las que llegar, se investigó en las tendencias en los últimos años de los consumidores a la hora de elegir un sistema operativo [26].

Operating System	4Q16 Units	4Q16 Market Share (%)	4Q15 Units	4Q15 Market Share (%)
Android	352,669.9	81.7	325,394.4	80.7
iOS	77,038.9	17.9	71,525.9	17.7
Windows	1,092.2	0.3	4,395.0	1.1
BlackBerry	207.9	0.0	906.9	0.2
Other OS	530.4	0.1	887.3	0.2
Total	**431,539.3**	**100.0**	**403,109.4**	**100.0**

Source: Gartner (February 2017)

Figura 18: Tabla de ventas y porcentaje de mercado de móviles.

Como ya se ha dicho antes en cuanto al número de ventas de Android frente IOS, Android es el ganador con diferencia, llegando a superar a su eterno rival en más de 3,5 veces. Si a esto se le relaciona con el dato mencionado del número de usuario de un aparato móvil previstos para este año (5.000 millones de usuarios), la interpolación de datos nos da una cifra de alrededor de 4.000 millones de móviles con sistema Android. Pese a ser solo una estimación, ya son cifras muy considerables a la hora de decantarse por elegir el sistema, incluso si fuesen completamente erróneas, por razones externas (guerras, epidemias, catástrofes naturales); seguirá siendo muy superior a las de IOS.

Al tratarse de un proyecto de ingeniería, como se es de esperar, la opción de ir por el que más mercado tiene es la obvia, ya que al llegar a más gente hay más probabilidades de que se descargue más veces, reduciendo así el coste unitario de fabricación, y aumentando el margen de beneficios.

3.2.1 Android

En la actualidad se cree que Android es un sistema operativo relativamente nuevo en comparación con otros como podría ser Symbian (2011), sin embargo su historia viene de unos cuantos años más atrás, datando su existencia en el 2005 cuando era aún propiedad de Android Inc. No obstante, no fue hasta mediados del 2005 cuando Google, tras haberla apoyado económicamente; compró a Android Inc. Aunque su puesta en el mercado llegó en noviembre del 2007 y su primer lanzamiento de Apple pie (v.1.0) (primera versión de Android) el 22 de octubre de 2008 [27].

Android es un sistema operativo basado en el núcleo Linux. Convirtiéndose en los últimos años en el sistema operativo número uno para teléfonos inteligentes. Su expansión es inmensa, no quedándose solo en los dispositivos móviles o tabletas, sino también llegando a televisores, relojes e incluso coches.

Figura 19: Vehículo con sistema Android.

Uno de los puntos más fuertes de Android, y que desde la empresa destacan; es la flexibilidad que el sistema operativo ofrece y la sencillez que el consumidor encuentra para hacerse al teléfono. La plataforma es adaptable tanto para pantallas pequeñas y de baja resolución como para las de mayor resolución, contando con importantes tecnologías de conectividad útiles en el día a día como Bluetooth, Wi-Fi o 4G. Otras ventajas importantes que aporta Android son el gran soporte multimedia que posee, la amplia incorporación de funciones para cámara de fotos y vídeo, GPS, sensores para medir la aceleración, sensores térmicos y de presión, sensores fotoeléctricos o la fundamental capacidad de ser multitarea, es decir, que las aplicaciones que no se están ejecutando en primer plano reciben ciclos de reloj, o no se interrumpen [16].

En Android siempre han querido ir a la última moda o tendencia, si no es que la creaban ellos; innovando y creando con unas pautas muy bien definidas: la sencillez en la ejecución, visualmente atractivo y la originalidad, sin quitar claro la funcionalidad. Un buen ejemplo son sus diferentes versiones las cuales poseen el nombre de diferentes postres o dulces. En cada nueva versión que sale, el nombre del postre o dulce elegido empieza por una letra distinta, conforme un orden alfabético:

Letra	Nombre	Versión	Traducción
A	Apple Pie	v1.0	Tarta de manzana
B	Banana Bread	v1.1	Pan de plátano
C	Cupcake	v1.5	Magdalena
D	Donut	v1.6	Rosquilla o Dónut
E	Éclair	v2.0/v2.1	Pastel francés conocido como pepito o canuto
F	Froyo	v2.2	Abreviatura de “Frozen Yogurt”, Yogur Helado
G	Gingerbread	v2.3	Pan de jengibre
H	Honeycomb	v3.0/v3.1/v3.2	Panal de miel
I	Ice Cream Sandwich	v4.0	Sandwich de helado
J	Jelly Bean	v4.1/v4.2/v4.3	Gominola o judía de gominola
K	KitKat	v4.4	Kit Kat
L	Lollipop	v5.0/v5.1	Piruleta
M	Marshmallow	v6.0	Malvavisco o nube
N	Nougat	V7.0	Turrón

Tabla 3: Cronología y nomenclatura de las versiones de Android.

Figura 20: Representación de Android con las distintas versiones.

Android está considerado como uno de los modelos de negocio más prolíferos, pues su desarrollo estratégico contempla los factores que marcan tendencia en la sociedad y en el entorno de las empresas. Las herramientas y metodologías con las que cuentan y ofrecen están respaldadas por expertos de todos los ámbitos, ya sean en negocios, desarrolladores, marketing y estética, etc. Convirtiéndose así en poco tiempo en un modelo a seguir por desarrolladores de tendencia y competencia en el mercado.

Y es que Android, a diferencia de sus competidores para dispositivos móviles como iOS o Windows Phone, se desarrolla de forma abierta y se puede acceder tanto al código fuente como a una lista de incidencias donde se pueden ver problemas todavía no resueltos y reportar problemas nuevos. Acercando al usuario a ellos, pues con el código fuente y las herramientas que también ofrece Android, el usuario es capaz de cambiar cualquier aplicación que usa en el móvil, puede añadir widgets en la pantalla de inicio o implementar launchers (estilos o temas en las pantallas de inicio); tiene competa libertad para cambiar y remodelar a su gusto lo que los desarrolladores previamente han creado.

Tras estos datos no cabe duda de que Android sea capaz de superar cada año en la cuota de mercado a sus competidores más directos. Cabe destacar la cuantiosa afluencia de este sistema operativo en países tan pioneros y relevantes como Alemania, Francia, España, E.E.U.U. o China, no obstante, su cuota de mercado en países como Japón se queda por detrás, donde, aunque por poco; el sistema IOS sale vencedor [20].

3.3 Programas para la creación de apps

Una vez que se ha elegido el sistema operativo, falta saber con qué herramienta se va a desarrollar la app. En una primera pasada por los distintos desarrolladores que podemos encontrar para Android, los más conocidos y por ello más usados serían:

- Eclipse
- App inventor
- Android Studio
- Basic 4 Android

Lo primero en este caso es comprobar las características de cada programa, siendo de gran utilidad la opinión de expertos desarrolladores y de la información disponible en cuanto a tutoriales y ejemplos que se encuentran en la red.

Todas ellas son de gran renombre e historia, no quedándose ninguna atrás en la preselección, no obstante los matices y opciones que aporta cada una hacen ver realmente cómo de apropiadas son para el desarrollo de este Trabajo de Fin de Grado.

3.3.1 Eclipse

Eclipse es un software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones y proyectos. En un primer momento se barajó la opción de realizar la aplicación en este entorno de desarrollo principalmente de lenguaje Java, dado su extendido uso en la creación de aplicaciones [28].

Se trata de una potente y completa plataforma de programación, desarrollo y compilación de elementos muy variados. Entre las características más destacables de esta herramienta cabe destacar sus potentes funciones y auge entre los desarrolladores, teniendo así un gran cantidad de documentación especializada.

Figura 21: Logo de Eclipse.

El software se basa en una plataforma de cliente enriquecido, la cual está constituida por una plataforma principal, y unas características secundarias. Las funciones de Eclipse son más bien de carácter general, aunque se pueden ampliar e implementar mediante el uso de módulos (plug-ins).

Esto es un gran avance respecto a sus competidores, pues la concepción de módulos permite a Eclipse ser una plataforma ligera para componentes de software y además de posibilitar la extensión a otros lenguajes de programación como son C/C++ o Python. En el transcurso de sus diferentes versiones se ha ido mejorando la interfaz del programa y creando ayudas para los usuarios como atajos o simplificaciones en el código.

3.3.2 App Inventor

Google AppInventor en una plataforma desarrollada por Google Labs para crear apps para el sistema operativo Android. De forma visual, y a partir de un conjunto de herramientas básicas, el usuario puede ir enlazando y uniendo una serie de bloques para implementar la aplicación. Se trata de un sistema completamente gratuito y fácil de configurar desde su página web oficial [16].

App Inventor tiene un editor de bloques que utiliza Open Blocks (librería de Java para crear un lenguaje visual a partir de bloques). Estas librerías están distribuidas por el MIT bajo su licencia libre. El compilador que traduce el lenguaje visual a Android en la aplicación utiliza Kawa (lenguaje de programación).

Figura 22: Logo de App Inventor.

En julio del 2010 sale la primera versión dirigida principalmente a personas con poca familiarización en el campo de la informática. En sus primeros comienzos el software tuvo gran éxito y aceptación pero duró poco ya que se quedó atrás respecto a otros entornos de desarrollo en poco tiempo. No obstante, su vuelta al mercado no se hizo esperar mucho, cuando la compañía decidió unir esfuerzos con el prestigioso Instituto de Tecnología de Massachusetts (MIT).

Desde ese momento, el destino de esta herramienta ha cambiado de forma significativa, cuando se abrió el código y se lanzó una segunda versión más completa y sin tantos fallos.

3.3.3 Android Studio

Figura 23: Logo de Android Studio.

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece una vasta colección de funciones, aumentando la productividad durante la compilación de apps para Android. Algunos ejemplos son los siguientes [22]:

- Un emulador rápido con varias funciones.
- Un entorno unificado en el que se puede realizar desarrollos para todos los dispositivos Android.
- Integración de plantillas de código y GitHub para ayudar a compilar funciones comunes de las apps e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK.
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine.

Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

- Módulos de apps para Android.
- Módulos de bibliotecas.
- Módulos de Google App Engine.

La interfaz que el usuario va a ver al trabajar en un proyecto de Android Studio sería similar a la siguiente figura, no obstante; se le da al usuario la posibilidad de cambiar de lugar algunos comandos y de reubicar algunas tablas de herramientas para un mayor espacio de visualización si así quisiera.

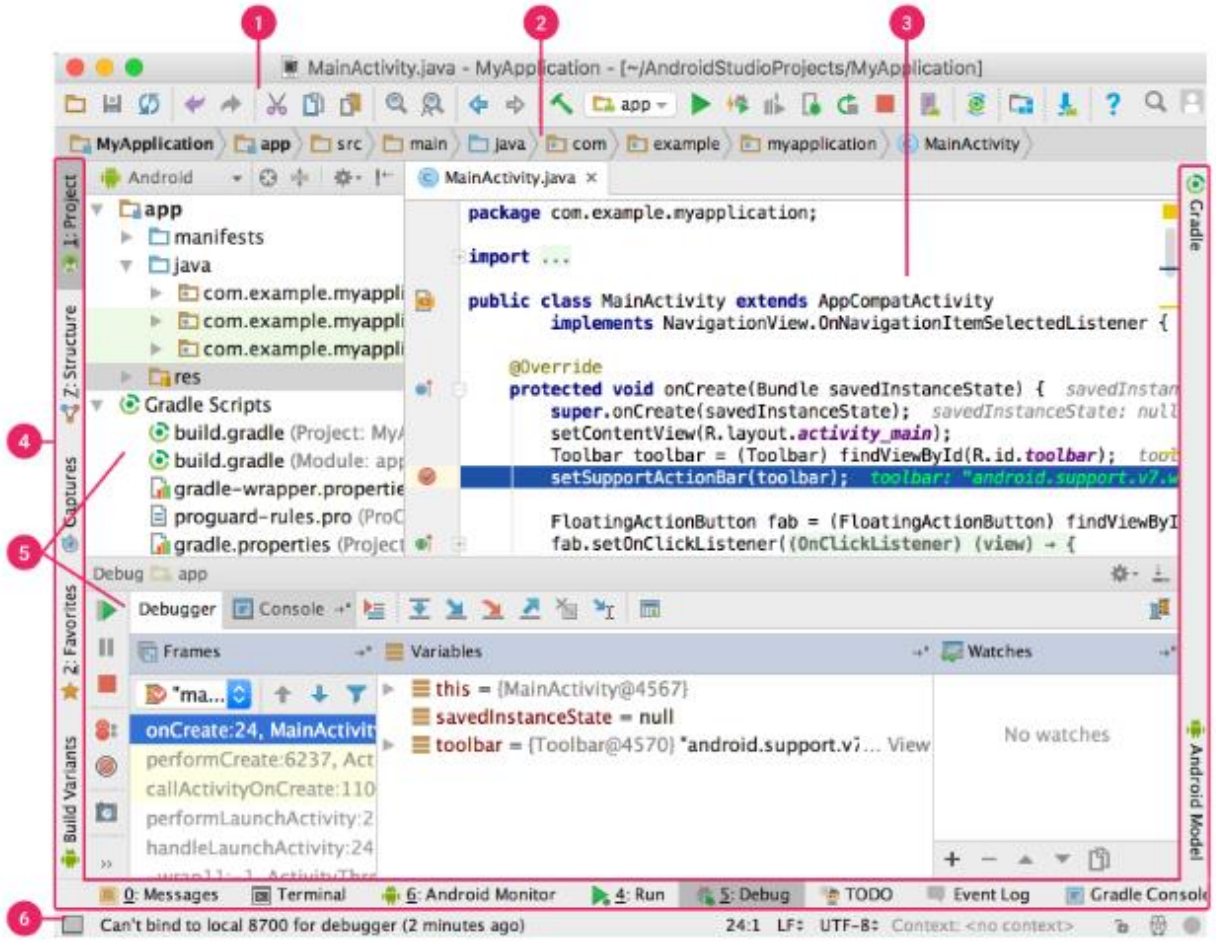

Figura 24: Pantalla principal en Android Studio.

1. La **barra de herramientas** te permite realizar una gran variedad de acciones, como la ejecución de tu app y el inicio de herramientas de Android.
2. La **barra de navegación** te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana **Project**.
3. La **ventana del editor** es el área donde puedes crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Por ejemplo, cuando se visualiza un archivo de diseño, el editor muestra el editor de diseño.
4. La **barra de la ventana de herramientas** se extiende alrededor de la parte externa de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.
5. Las **ventanas de herramientas** te permiten acceder a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc. Puedes expandirlas y contraerlas.
6. En la **barra de estado**, se muestra el estado de tu proyecto y del IDE en sí, como también cualquier advertencia o mensaje.

3.3.4 Basic 4 Android

Basic4Android es una herramienta de desarrollo rápido de aplicaciones para las apps nativas de Android, desarrollado y comercializado por Anywhere Software Ltd. Perfecta para principiantes en el mundo de la creación de apps sin experiencia previa y sin cocimientos de la lengua Java [29].

Figura 25: Logo Basic 4 Android.

B4A es una alternativa a la programación con Java y el SDK de Android, ya que no es necesario tener conocimientos en lenguaje Java, puesto que no se necesitan. Su diseñador visual simplifica el proceso de creación de interfaces de usuario que apuntan a teléfonos y tabletas con diferentes tamaños de pantalla. Los programas compilados pueden probarse en emuladores de AVD Manager o en dispositivos Android reales usando Android Debug Bridge y B4A Bridge.

El lenguaje en sí es similar a Visual Basic y Visual Basic .Net, aunque está adaptado al entorno nativo de Android. B4A es un lenguaje basado en objetos y en eventos. El software genera aplicaciones Android firmadas estándar que se pueden cargar en tiendas de aplicaciones como Google Play, Samsung Apps y Amazon Appstore.

3.3.5 Comparación y decisión sobre la plataforma a elegir

A la hora de eliminar las sucesivas opciones de la lista hasta quedarse solo una, se tuvo en cuenta los factores ya mencionados antes como la simplicidad a la hora de crear la app, como fuese de amigable la interfaz del programa, la potencia y capacidad del mismo, el soporte y ayudas que se encuentran a disposición de los usuarios, etc.

La primera opción en ser descartada fue Basic 4 Android por su falta de funciones y complejidad. Pese a ser muy sencilla de usar y reducir en gran medida los tiempos en la creación de apps, al ausencia de funciones más completas hacía inviable el proyecto, sentenciándola así fulminantemente.

La siguiente en desaparecer de la lista fue App Inventor por razones más o menos similares. Pese a tener un mayor número de aplicaciones a disposición, unas librerías estupendas y una complejidad superior a la de Basic 4 Android cojea de un pie. Al indagar más sobre las ventajas e inconvenientes que ofrecía el software, se encontraban limitaciones a la hora de ejecutar varios comandos a la vez o tener múltiples pantallas, las cuales estaban capadas a 10; apareciendo el error de posible error de compilación.

Siendo sin embargo una buena opción no se adapta a las necesidades de este proyecto, haciendo, como a la opción previa; que se tenga que descartar para llevar a cabo la creación de la app propuesta.

Por último nos quedan solo dos opciones, Android Studio y Eclipse. Para este último asalto en la búsqueda del software más apropiado para el caso que se presenta en este proyecto, se puso en el rin a las dos opciones, midiendo esta vez el número de recursos disponibles de cada uno.

Eclipse tiene a sus espaldas una larga y buena reputación como plataforma para crear apps, eficiente, compleja, versátil; un muy buen programa al fin y al cabo. Su rival, Android Studio; no se queda atrás, siendo más nueva y actualizada que Eclipse cuenta con un mayor número de referencias y ventajas; haciendo que incluso los usuarios de Eclipse se pasen a la competencia. Este factor que se ha podido observar en diferentes páginas web y blogs sobre creación de apps, se ha ido repitiendo y aumentando a medida que Android Studio cogía más y más fuerza. Estando a un mismo nivel de complejidad a la hora de generar código Java, Android Studio emerge vencedor por la mayor afluencia de personas usándola, una cantidad

superior de sitios donde buscar información y tutoriales para su uso y una interfaz y código que actualizan constantemente cada corto periodo de tiempo.

Como prueba de este último dato, durante la elaboración de este Trabajo de Fin de Carrera, se tuvo que actualizar el programa en si tres veces, mejorando en cada una de ellas la accesibilidad de los comandos y el número de librerías disponibles, y una actualización de los sistemas SDK y AMV cada dos o tres semanas para purificar los errores que la gente se iba encontrando y los reportaba.

Tras esto último podemos dar como vencedor a Android Studio, siendo este el programa con el que se realizara este proyecto.

3.4 Funcionamiento de Android Studio

Como se venía diciendo anteriormente, y tras haber debatido cuál era la mejor opción, el programa a utilizar para el desarrollo de la aplicación seria Android Studio. Emprenderse directamente con la app de volantes de inercia sería un poco lanzarse a una piscina desde el trampolín sin saber si quiera si tiene agua, por lo que unos primeros pasos de reconocimiento por el programa y una mini aplicación de testeo será lo más recomendado.

Para empezar un proyecto en Android Studio basta con iniciar el programa y elegir el tipo de proyecto que se quiere desarrollar, dando a elegir entre una numerosa cantidad de opciones para que el usuario escoja la que más se adecue a su proyecto.

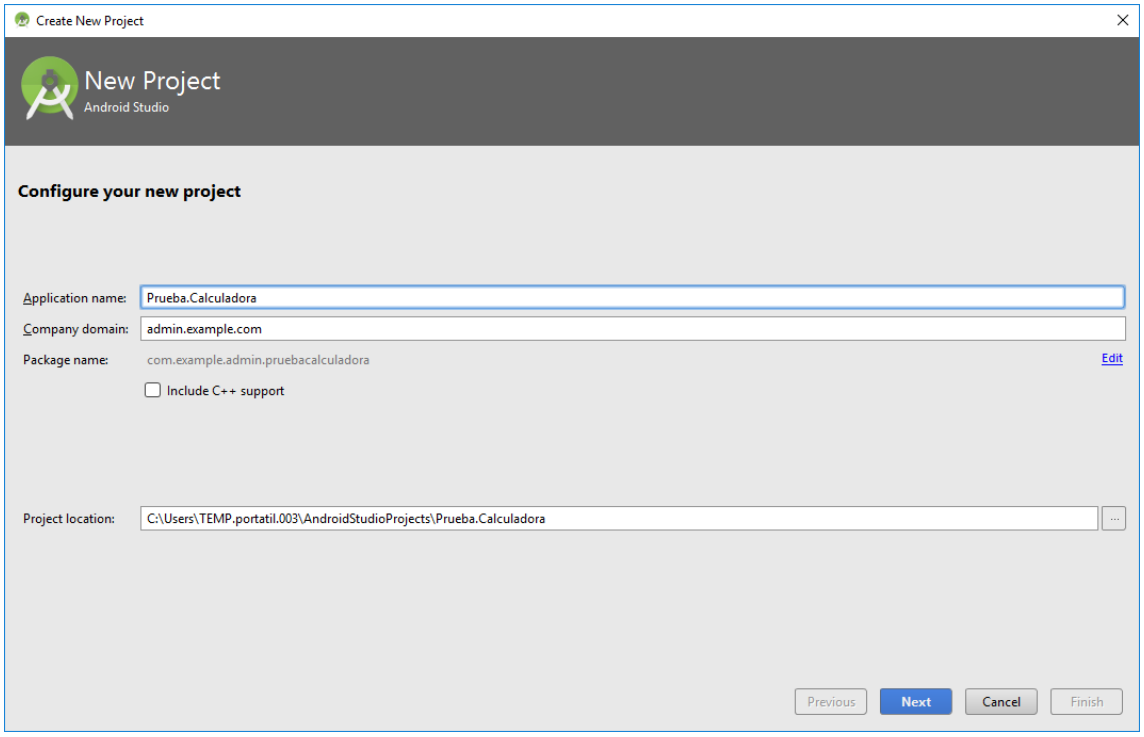

Figura 26: Pantalla para crear un nuevo proyecto en Android Studio.

Unas de las características buenas que tiene Android Studio es la de incluir soporte para lenguaje C++ en caso de querer importar proyectos de otro sistema al de Java. Eligiendo el directorio del proyecto y el nombre de la aplicación se empieza a dar los pasos para su creación.

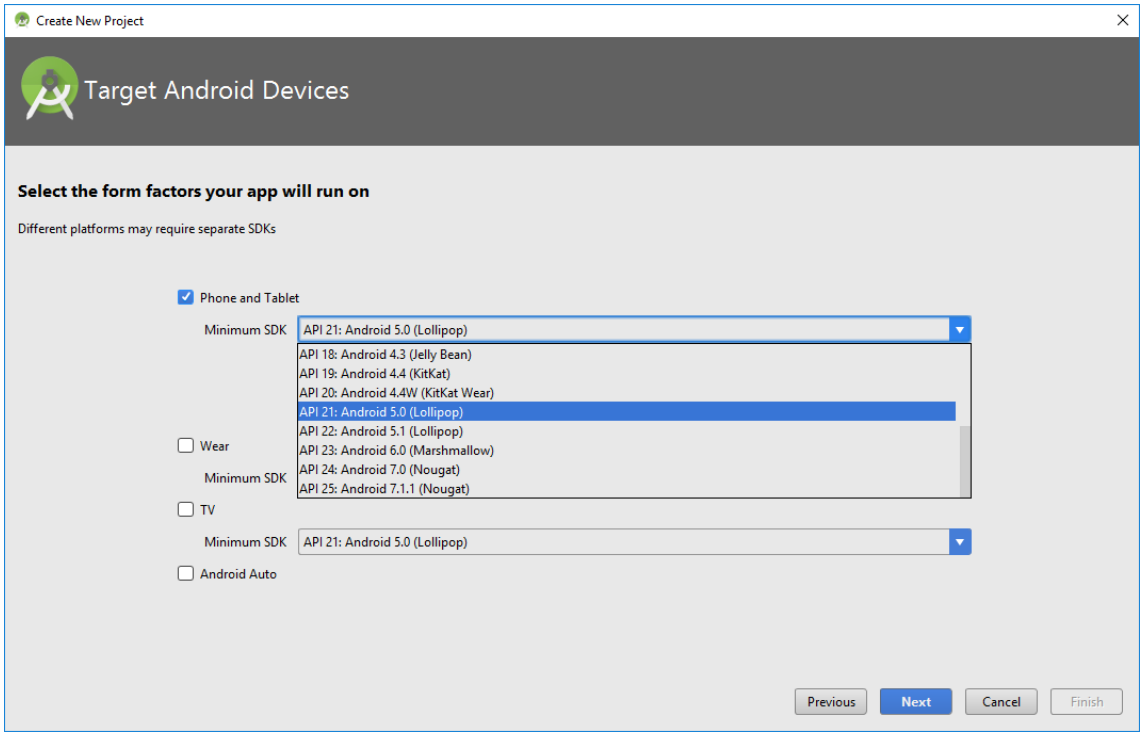

Figura 27: Selección de datos técnicos para un proyecto en Android Studio.

En la siguiente pantalla se ofrece una serie de opciones técnicas como la de seleccionar el SDK. El SDK viene a fijar la versión mínima que tiene que tener el aparato para poder reconocer y correr el programa. Como esto depende del número de aparatos que hay en el mundo actualmente, el número de versiones que hay para Android disponibles y la calidad de las funciones que están permitidas para cada versión se puede hacer un poco complicado su elección. Por ello el mismo Android Studio ayuda al usuario con la selección, dando como predeterminada la API 21 Android v5.0 Lollipop, no obstante; da la posibilidad de cambiarlo a una versión mayor o menor, incluyendo un gráfico del número de usuarios y características de cada uno.

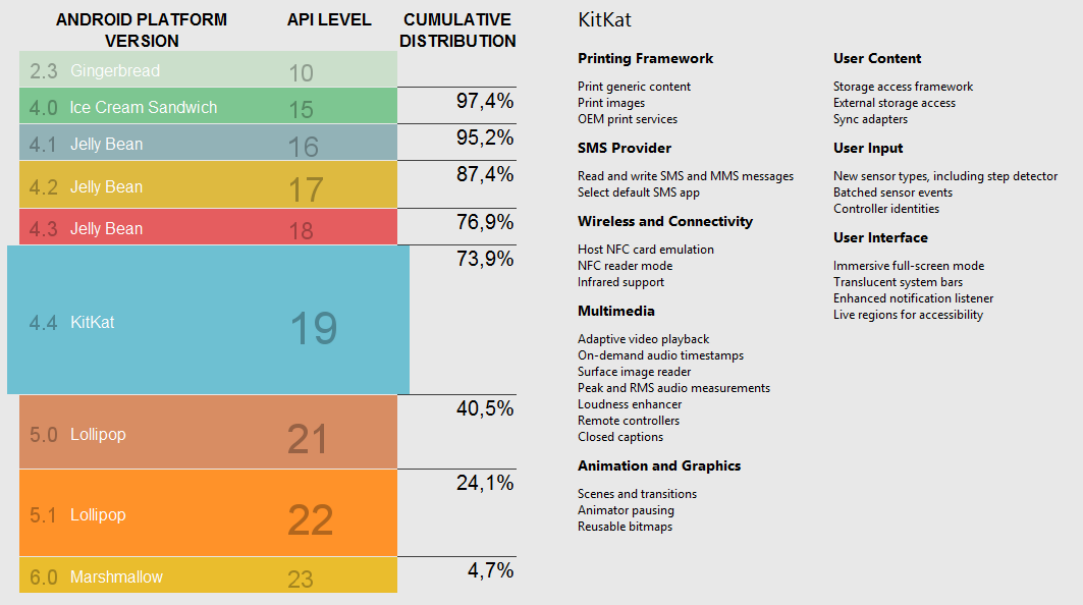

Figura 28: Proporción de usuarios según la versión de Android.

Como se puede deducir de los datos, la versión 4.4 KitKat es la más abundante dentro de lo nuevo, sin embargo no es la más recomendada ya que el número de funciones y su implementación en el código no es muy buena, dando errores de librería o inhabilitando funciones. Por ello, y porque el mismo Android Studio recomienda Android v5.0 Lollipop, el programa se desarrollará en esa versión.

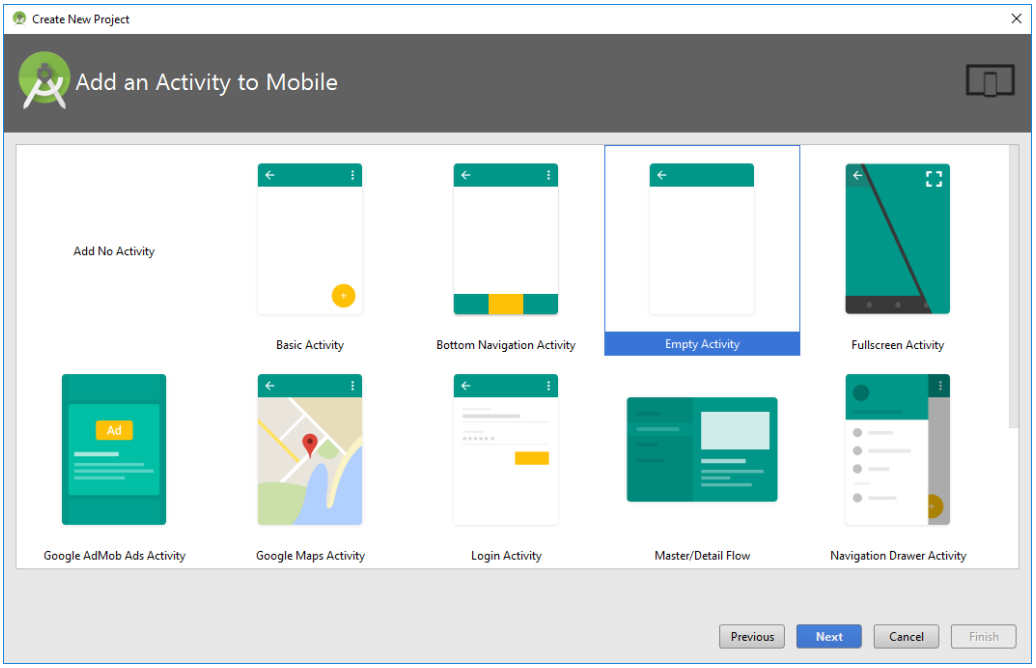

Figura 29: Selección del tipo de plantilla en Android Studio.

Una vez elegido los parámetros más técnicos de la aplicación, hay que elegir el tipo de pantalla que se utilizará como plantilla. De entre todas las opciones la más cómoda de usar es la Empty Activity, con ella se creará la app de prueba *Calculadora* y la app final de *CaVI (Cálculo de volantes de inercia).*

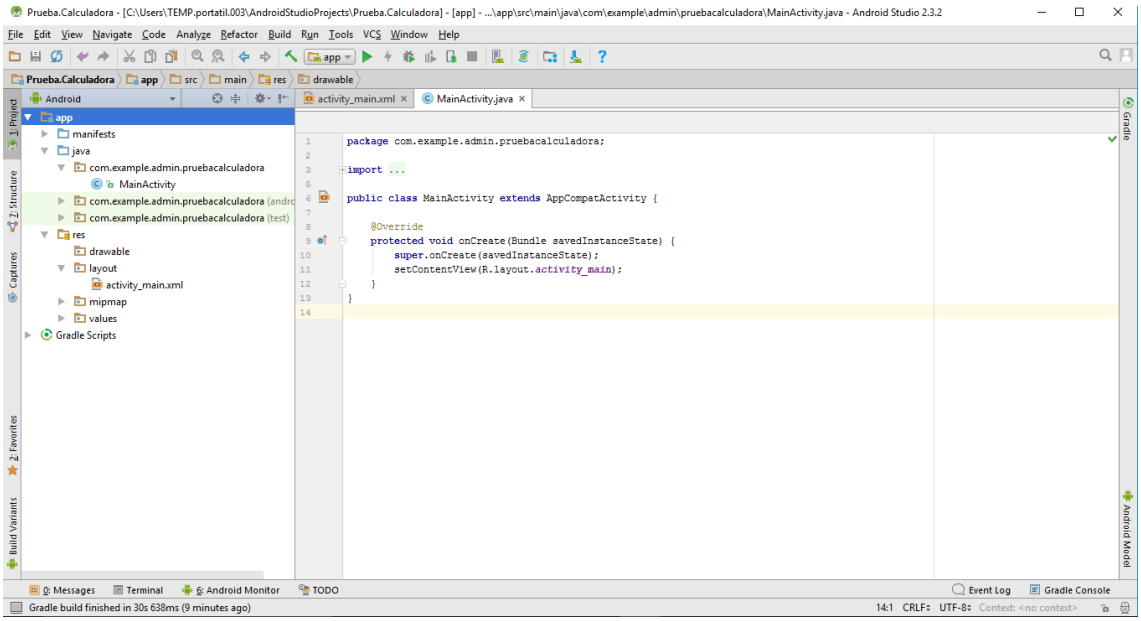

Figura 30: Pantalla principal del proyecto en Android Studio.

Con la actividad abierta se puede ver toda la interfaz principal de la aplicación. Como ya se comentó en apartados posteriores, desde aquí se hace todo tipo de operaciones. Una cosa que no se mencionó pero que es de remarcar es la posibilidad de importar bibliotecas. Esto permite al desarrollador acceder a funciones, diseños u opciones más sofisticadas que las predeterminadas por Android Studio. En el desarrollo de la calculadora no se usó ninguna librería extra, sin embargo para la app de cálculo de volantes de inercia se importó una librería específica para la implementación de gráficas.

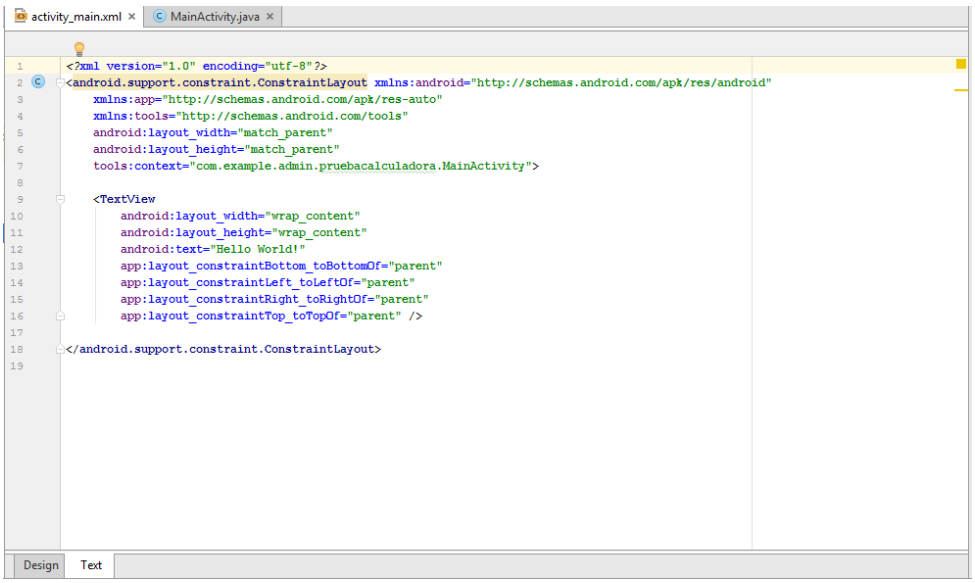

Figura 31: Visualización de la pantalla en código.

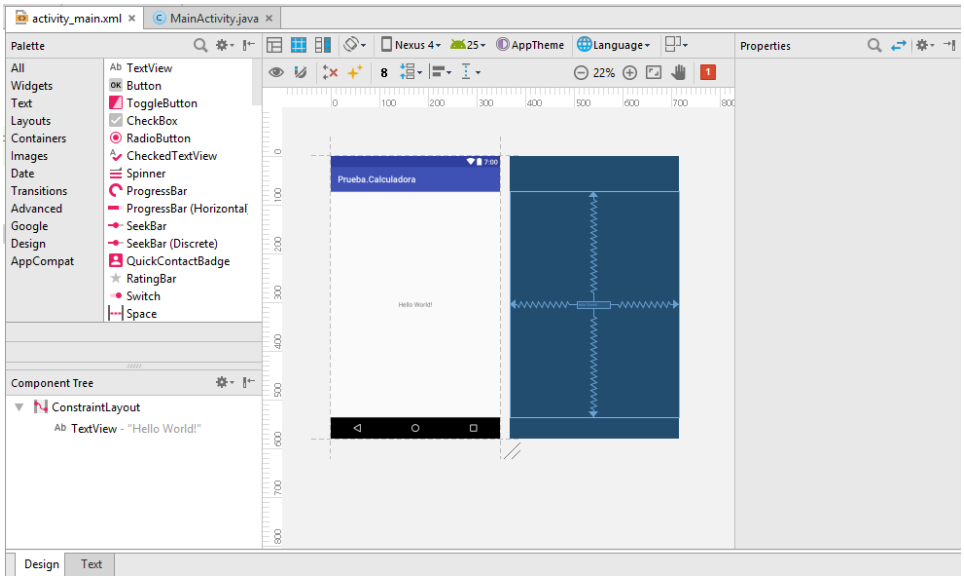

Figura 32: Visualización de la pantalla como si fuese un dispositivo.

Estas dos pantallas, por muy distintas que parezcan, son la misma. La primera es el modo del código donde aparece cada comando, formando el conjunto para el diseño de cada entrada. Mientras que la segunda es el modo de visualización, donde se colocan los apartados (entradas de texto, botones, fotos…). Al poner un nuevo widget en una de las dos pantallas se actualizará inmediatamente la otra, siendo una traducción de código a visual o, visual código.

Para las opciones de visualización, se puede observar en la siguiente figura a la izquierda, en el apartado *Palette,* todas las opciones disponibles para añadir a la pantalla. Botones, visualización de texto, texto interactivo, diseños para fijar el resto de opciones, barras para deslizar la pantalla, etc. De ahí se cogerán todos los comandos necesarios para integrarlos en la aplicación.

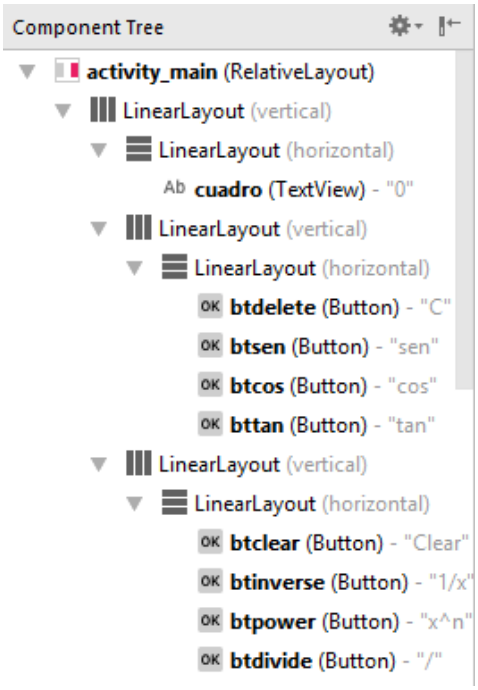

Figura 33: Árbol de componentes.

Con la aplicación Calculadora ya escrita, se puede comprobar como ya se tienen más componentes en la barra de Árbol de componentes (component tree). Empezando por la actividad principal, se va bajando diseño a diseño desglosando todo lo que tiene la aplicación. Cada componente hay que nombrarlo distinto y registrarlo dentro de la aplicación para que luego pueda reconocerlo cuando corra en el aparato móvil o simulador.

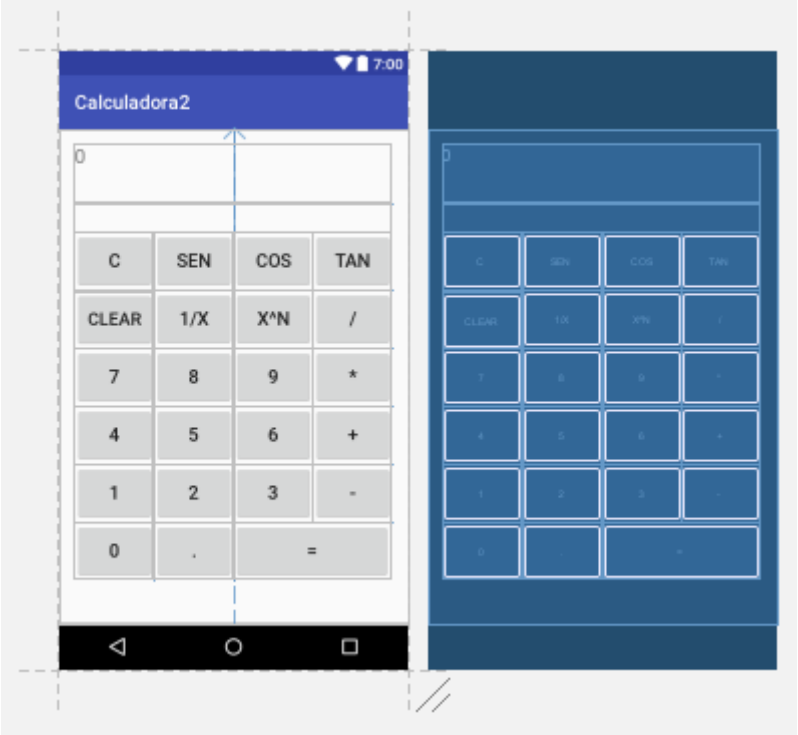

Figura 34: Diseño de la aplicación “Calculadora”.

Al final de todo el código la aplicación Calculadora tendría esta apariencia. Los botones numéricos, las operaciones de suma, resta, multiplicación, división, seno, coseno, tangente, inverso y potencia. Dando la posibilidad de borrar número a número o d eliminar todo.

```
RelativeLayout | LinearLayout | LinearLayout | LinearLayout | Button
        </LinearLayout>
    </LinearLayout>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:text="0"
                android:id="@+id/btcero"
                android:layout_height="60dp"
                android:onClick="cero"
                android:textSize="19sp"
                android:layout_width="82dp" />

            <Button
                android:text="."
                android:id="@+id/btdot"
                android:layout_height="60dp"
                android:onClick="punto"
                android:textSize="19sp"
                android:layout_width="82dp" />
```

Figura 35: Código de la visualización de la aplicación "Calculadora".

Lo que antes se veía en forma visual, ahora esta traducido en forma de código.

```
MainActivity | equal()
10
11
12  public class MainActivity extends AppCompatActivity {
13
14      Button btuno, btdos, bttres, btcuatro, btcinco, btseis, btsiete, btocho, btnueve, btcero,btdot,btsen,btcos,bttan,btequal,btinv
15      TextView cuadro,concatear;
16      Integer operador,mm;
17      double num1, num2, resultado;
18      @Override
19      protected void onCreate(Bundle savedInstanceState) {
20          super.onCreate(savedInstanceState);
21          setContentView(R.layout.activity_main);
22
23
24          btcero=(Button)findViewById(R.id.btcero);
25          btuno=(Button)findViewById(R.id.btuno);
26          btdos=(Button)findViewById(R.id.btdos);
27          bttres=(Button)findViewById(R.id.bttres);
28          btcuatro=(Button)findViewById(R.id.btcuatro);
29          btcinco=(Button)findViewById(R.id.btcinco);
30          btseis=(Button)findViewById(R.id.btseis);
31          btsiete=(Button)findViewById(R.id.btsiete);
32          btocho=(Button)findViewById(R.id.btocho);
33          btnueve=(Button)findViewById(R.id.btnueve);
34          btdot=(Button)findViewById(R.id.btdot);
35          btsen=(Button)findViewById(R.id.btsen);
36          btcos=(Button)findViewById(R.id.btcos);
37          bttan=(Button)findViewById(R.id.bttan);
38          btequal=(Button)findViewById(R.id.btequal);
39          btinverse=(Button)findViewById(R.id.btinverse);
40          cuadro=(TextView)findViewById(R.id.cuadro);
```

Figura 36: Código para la asignación de variables.

Después de dar el diseño y visualización al programa hay que darle las tripas. El groso del código se encuentra en los archivos tipo .java donde se crea la relación de los distintos componentes entre sí. Como se dijo previamente, hay que asignar un nombre a cada componente e identificarlo para que el programa los reconozca. En la figura anterior se ve cómo se han nombrado los botones primero y luego se los ha ido poniendo uno a uno para que la aplicación los tenga reconocidos.

```
44      public void one(View v) {
45          concatear=(TextView)findViewById(R.id.cuadro);
46          cuadro.setText(concatear.getText().toString()+"1");
47
48      }
49      public void dos(View v)
50
51      {
52
53          concatear=(TextView)findViewById(R.id.cuadro);
54          cuadro.setText(concatear.getText().toString()+"2");
55
56      }
57      public void tres(View v)
58
59      {
60
61          concatear=(TextView)findViewById(R.id.cuadro);
62          cuadro.setText(concatear.getText().toString()+"3");
63
64
65      }
66      public void cuatro(View v)
67      {
68
69          concatear=(TextView)findViewById(R.id.cuadro);
70          cuadro.setText(concatear.getText().toString()+"4");
71
```

Figura 37: Asignación de las funciones a los botones.

Con todos los botones y visualizaciones de texto reconocidas, se empieza a codificar qué tiene que hacer cada uno al ser pulsado. En los ejemplos que se aprecian en la figura anterior, se ve cómo se ha asignado al botón “1” que al ser pulsado añada un 1 a la visualización de texto.

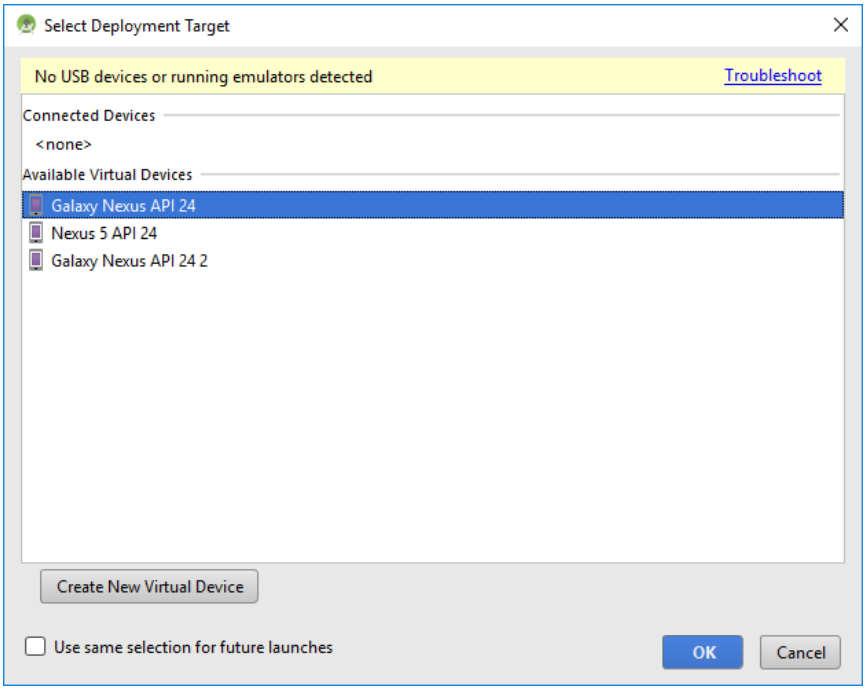

Figura 38: Selección del aparato para la simulación virtual.

Una vez que se ha asignado una función a cada botón, se puede dar por concluida la aplicación y se dispone a probarla. Para ver si funciona sin errores antes de pasarla a un teléfono real, Android Studio brinda la oportunidad al usuario de testarla en un simulador virtual. En el simulador hay infinitas posibilidades de aparatos, desde teléfonos hasta tablets y televisores inteligentes. Pudiendo elegir la versión del teléfono y el tamaño de la pantalla se eligen varios para probar que no se descuadra en ninguno de ellos.

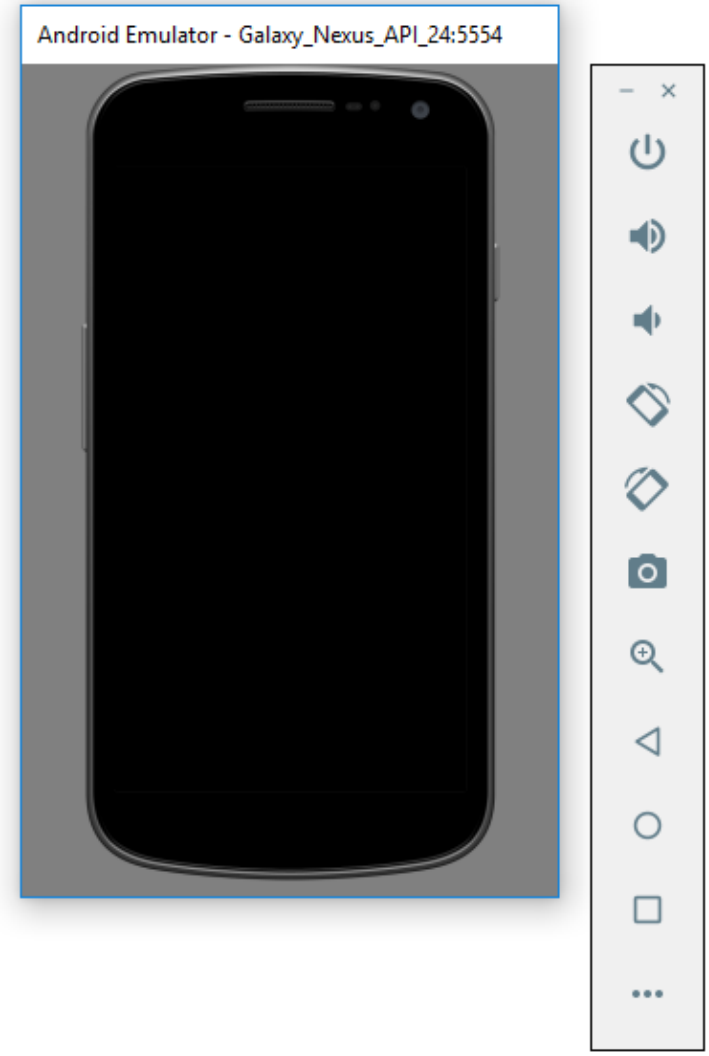

Figura 39: Apariencia del dispositivo virtual.

Figura 40: Apariencia de la aplicación "Calculadora" en el dispositivo virtual.

Con el simulador se comprueba que la aplicación corre perfectamente y ninguno de los botones se ha descuadrado. También da al usuario la oportunidad de girar el móvil, pulsar el resto de botones y prácticamente utilizar el simulador como un móvil real.

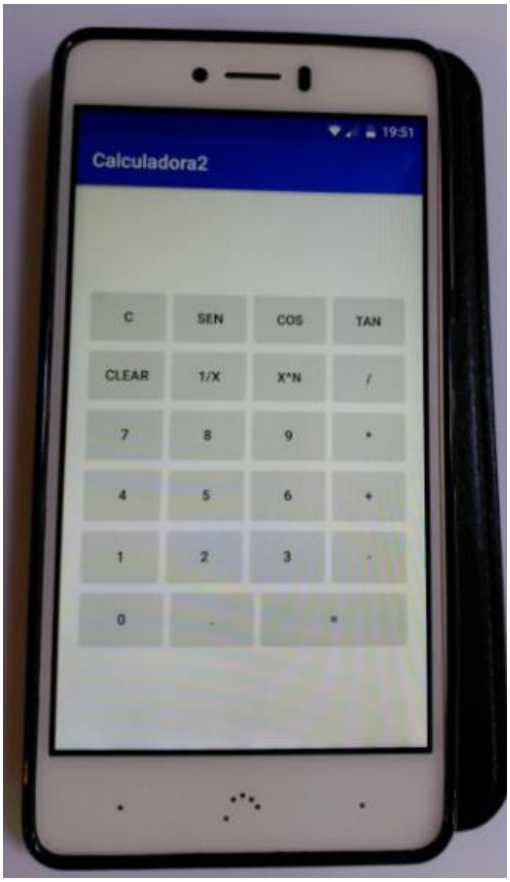

Figura 41: Aplicación "Calculadora" probada en un móvil físico.

Sabiendo que la aplicación no tiene problemas funcionales ni cambia el diseño se prueba en un aparato móvil real. En este caso un Aquaris U5 BQ versión 6.0.

Algunas anotaciones que se pueden hacer son los fallos en la integración de la aplicación. Al tratarse de una aplicación de prueba y no tener unos medios muy amplios en la creación de las mismas, se encuentran errores como apretar el botón "=" sin ningún numero puesto y saltar el error "La aplicación Calculadora se ha detenido. Quiere reiniciar la aplicación". Esto viene por la creación de un botón que actúa un cuadro de texto sin tener un verificador de que haya un número primero. Pese parecer muy obvio de que no se le debe dar al botón igual antes de tener un número, puede pasar sin querer, pero el usuario solo entendería que la app se ha parado, pudiendo pensar que esta defectuosa.

Marcar varias veces el comando "coma" sí que se restringió dado que un clic de más podría conllevar a un cierre de aplicación indeseado. Para las siguientes apps a crear se tendrán en cuenta todos los error vistos en esta aplicación, a fin de intentar eliminar todas los

posibles acciones que hiciesen cerrar el programa y hacer pensar al usuario que la aplicación que ha descargado es defectuosa.

Dando por zanjada y satisfactoria la creación de la app de prueba “Calculadora”, se indagan más en opciones de comandos y formas de crear múltiples actividades (pantallas), ya que serán necesarias para la app final. Además de estos conocimientos extra, se requieren mini pruebas para saber, en el caso de error; dónde se ha originado. Todas estas pruebas y adquisición de conocimientos se contabilizan en el tiempo de desarrollo de la aplicación final, ya que una vez se tenga el código correcto y se entienda su funcionamiento, se copiará y colocará en su debido lugar para no tener que meter todo desde cero.

Haciendo lo que sería un colage de mini aplicaciones se creará la app final, siendo así muy útil Android Studio por su versatilidad de exportar o importar proyectos dentro unos de otros, evitando tener que reescribir todo el código desde sus cimientos. Esto ahorra una ingesta cantidad de tiempo, para hacerse una idea de las líneas que tiene la prueba de la calculadora, está sola app llega a casi las 300 líneas de código java, más otras casi 200 de diseño.

En la nueva aplicación se estima que estos números se incrementen exponencialmente, pudiendo llegar a más de 400 líneas de código por actividad aproximadamente dado su complejidad y extensión. En la disertación del mismo que se hará aquí, solo se comentará el resultado final de la aplicación como si fuese hecha de cero, no se comentarán mini programa a mini programa, no obstante; se dedicará un tiempo a explicar cómo se llegó a cada función.

3.5 Funcionamiento de Sangit y Sangit Editor

Otro de los programas utilizados en este proyecto es Snagit. Este es un programa de captura de pantalla, y junto con su Snagit Editor permite el retoque de imágenes. Se podría hacer un símil con el programa Photoshop o Paint, ya que también ambos permiten el retoque de imágenes y fotos. Para poner en perspectiva dónde estaría, en cuanto a nivel profesional y de utilidad; Snagit se podría colocar en mitad de ambos, decantándose más hacia el lado de Photoshop.

El programa ayuda de manera muy simple la forma de coger imágenes o capturas de pantalla de cualquier cosa que se encuentre en la pantalla del ordenador. No solo permite la captura en imágenes sino que también en formato de vídeo. Esta última opción no es usada aquí pero no deja de ser un gran bonus para elegir este programa.

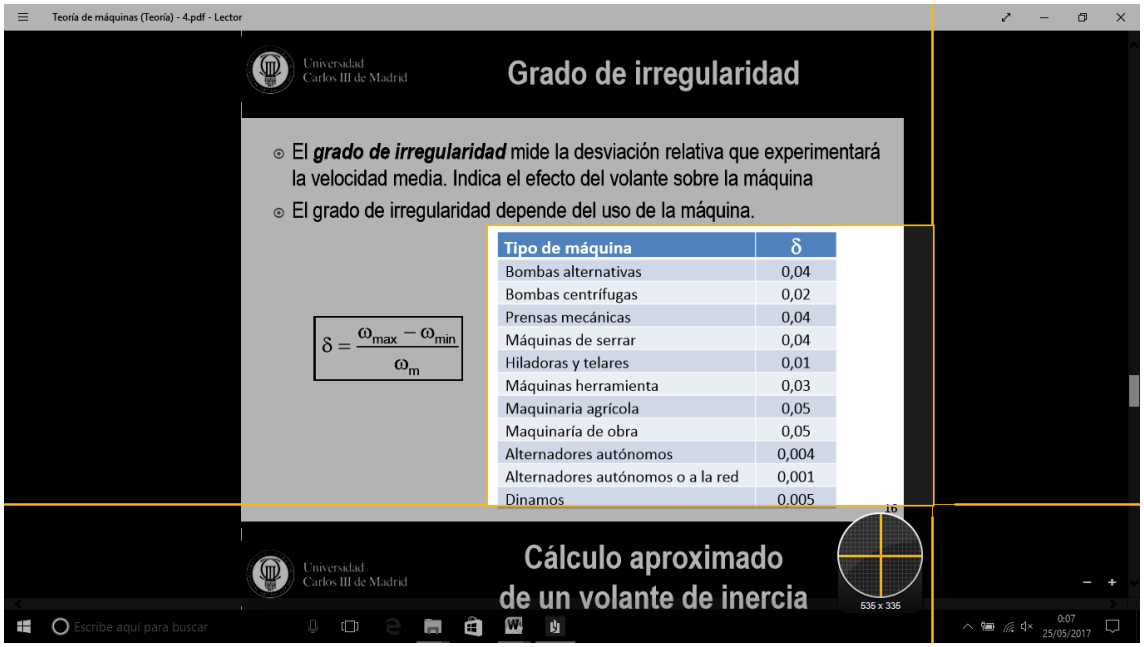

Figura 42: Modo de capturar una pantalla.

Apretando la tecla “imprimir pantalla (impr pa)” saldría lo que se muestra en la figura anterior. La pantalla se obscurecería, aparecería una cruz donde se situaría el ratón, la cual se moverá donde vaya el cursor, y haciendo clic izquierdo y manteniéndolo apretado, se podría recuadrar lo que se quiere seleccionar. Además de la cruz para poder saber las delimitaciones de la captura, aparecerá un círculo que ampliará los pixeles cerca del cursor para afinar más a la hora de encuadrar.

Si se suelta el botón izquierdo se cierra este modo y se pasa directamente a Snagit Editor donde se continuará ahora con las modificaciones que se le quiera hacer a la imagen cogida. Si se quisiera coger una imagen muy larga, la cual no entrase en la pantalla, aparecerían unas flechas indicando para poder bajar y coger todo.

Una vez en Snagit Editor se decide qué hacerle a la foto, ya sea añadir indicaciones, textos, flechas, juntarla con otras imágenes, mitigar el brillo, vaciar el fondo, pintar las imágenes, etc. La cantidad de opciones disponibles es muy amplia.

Para ilustrar el proceso de cómo se usa Snagit y Snagit Editor, se pondrá un ejemplo a continuación acompañado de imágenes.

Figura 43: Foto preparada de un elemento móvil del motor, tomada en un taller.

Con una imagen propia cogida por una cámara de fotos, se recorta con Snagit para que se ajuste a las medidas deseadas, quitando en la medida de lo posible el fondo. Una vez se envía a Snagit Editor, se utiliza el comando borrar, eliminando el fondo para dejar solo la figura.

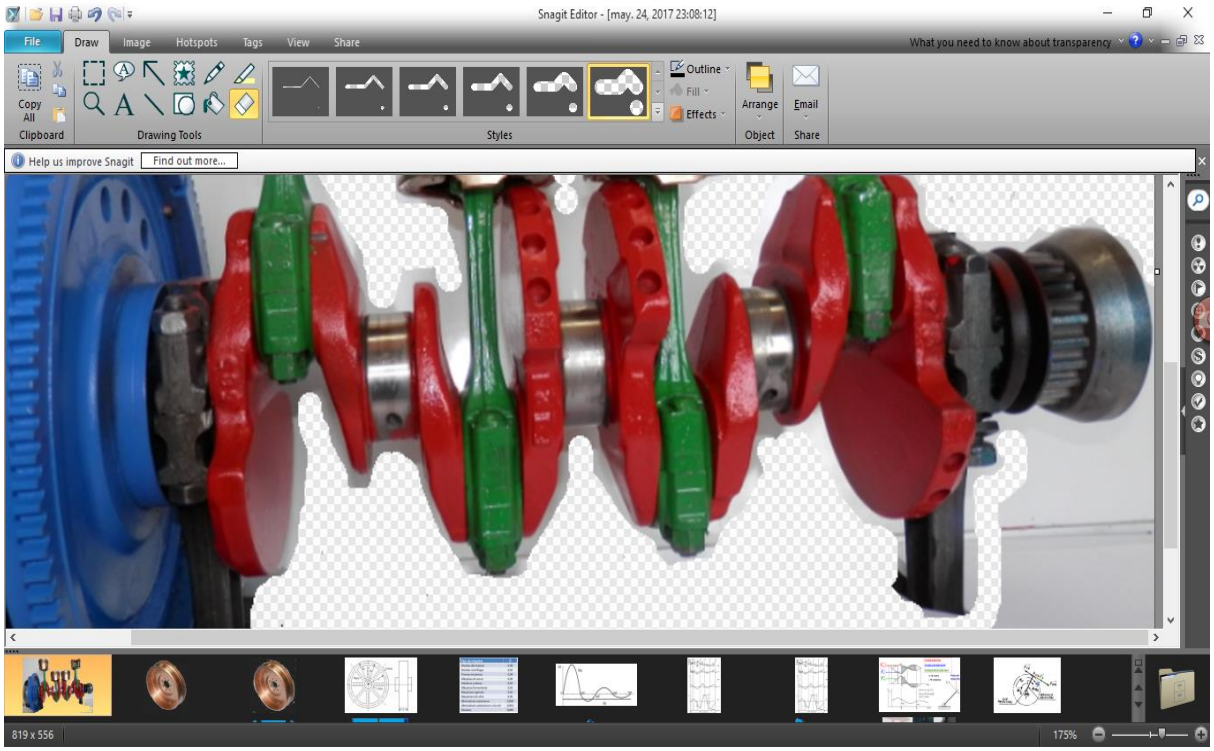

Figura 44: Uso del programa Sangit Editor.

En la parte inferior se aprecian las distintas imágenes que se han ido cogiendo con Snagit, en el centro la imagen que se está procesando, y arriba, las diferentes opciones de herramientas. Para el acabado de la pieza se utilizaron distintos grosores para meterse en la imagen lo menos posible. Finalmente quedaría la siguiente imagen una vez quitado todo el fondo.

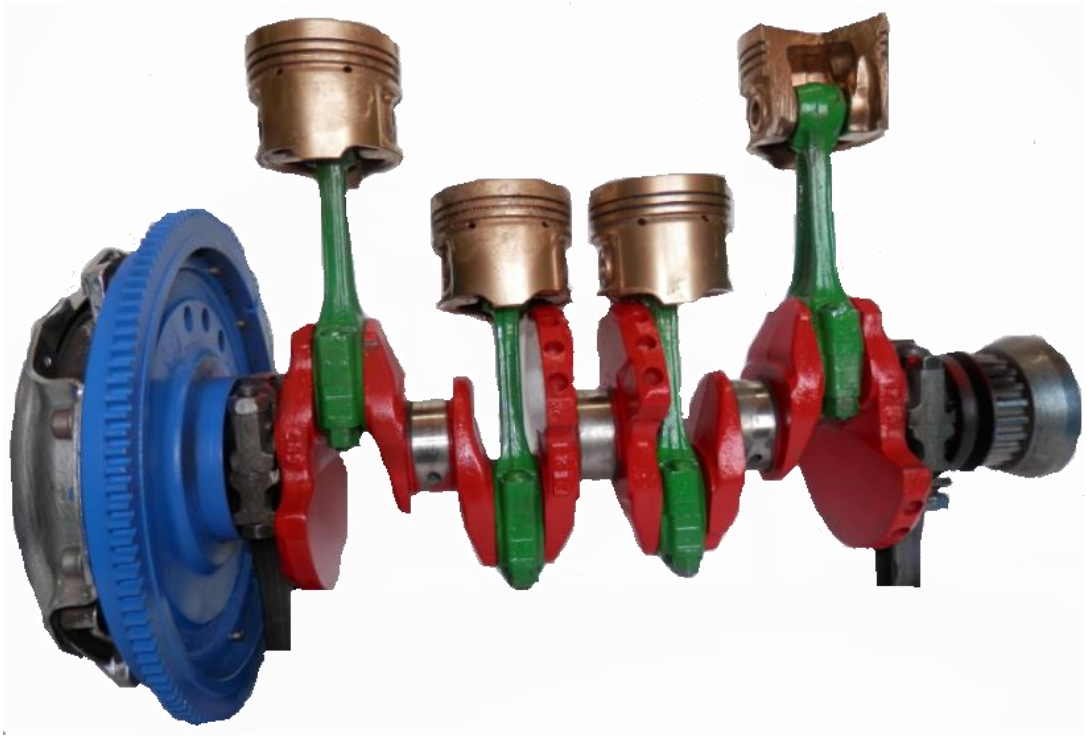

Figura 45: Imagen final después de haber sido tratada en Snagit Editor.

3.6 Funcionamiento de Solid Edge

Una de las herramientas usadas para el desarrollo de este trabajo de fin de grado es Solid Edge. Este programa CAD/CAM es extensamente usado para el diseño de piezas y conjuntos en 3D al igual que para toda índole de simulaciones en las últimas versiones del mismo.

Solid es un programa muy completo, teniendo una interfaz muy amigable y fácil de acostumbrarse a ella, permitiendo al usuario, por poco experimentado que sea en el diseño de piezas; crear modelos 3D de gran complejidad. Ya no decir, de aquellos que gocen de conocimientos sobre programas similares, los cuales lo encontrarán tremendamente útil.

Figura 46: Logo de Solid Edge.

Empezando desde la elección del tipo de proyecto que se quiere desarrollar, ya sea isométrico, pieza 3D, conjunto, chapa, etc. Se irá, operación a operación, conformando la pieza. Como primer paso se recomienda hacer un boceto, con el que luego se podrá extruir, lo que hará que cobre volumen. Después ya depende de la pieza que se desee crear, pues podría seguirle un vaciado, una simetría, otra extrusión, una operación por superficie, etc.

Dado el alto grado de maniobrabilidad que brinda este programa, se utilizará para la creación de los logos de los botones, remplazando las letras por una imagen que resemble la palabra, haciendo así que la app sea más atractiva visualmente para el usuario.

Uno de los ejemplos que se podrían dar para ilustrar el uso de Solid será el logo para el botón de dimensionar el volante.

Como se dijo anteriormente, esta figura se empezó por un boceto consistente en un círculo; aquí las medidas no son extremadamente necesarias, ya que lo que importa es el acabado visual (en cualquier otro caso o para un diseño de una pieza real, las medidas son fundamentales; pero al tratarse solo de hacer un dibujo, pasan a segundo plano).

Una vez hecho el primer círculo, se empieza a hacer vaciados, ya sea para las hendiduras de los engranajes como para las diferentes secciones del mismo. Repitiendo la misma operación pero con diferente forma se llegaría a un paso intermedio tal que el siguiente.

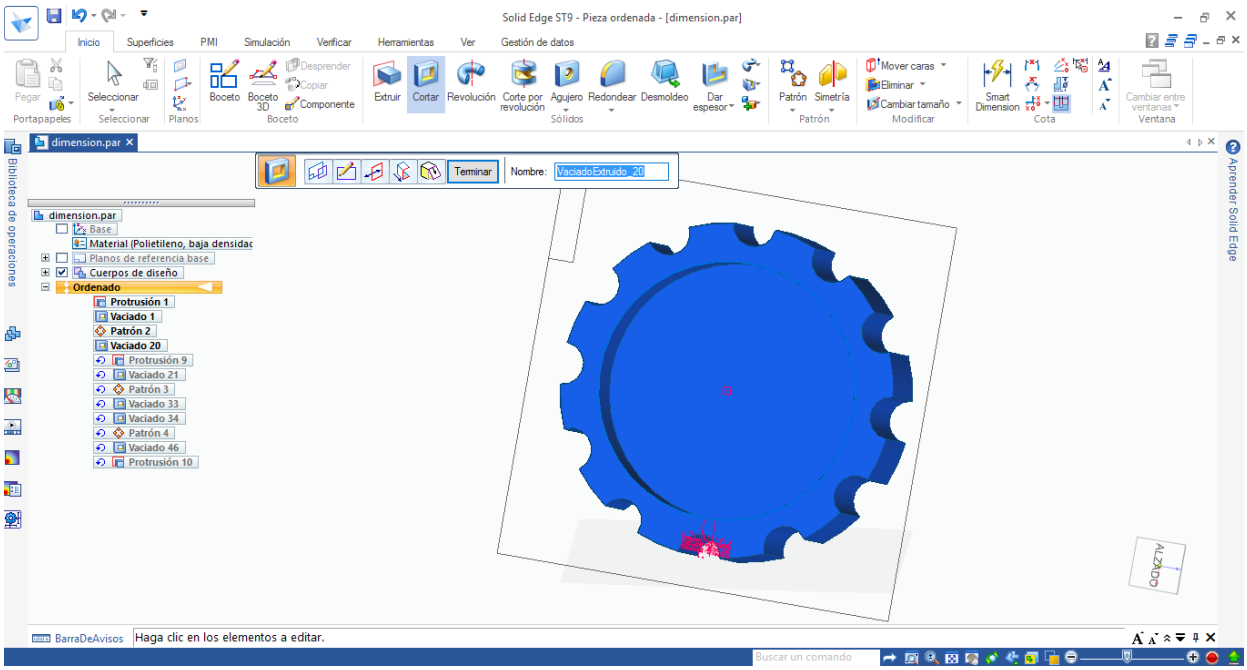

Figura 47: Diseño preliminar del logo.

Como se observa de la figura, se tiene aún la pieza basta de la que se irá quitando material para poder llegar a la versión final de la misma. A la izquierda se puede ver el listado de operaciones en un orden descendente. Arriba, la barra de operaciones disponibles y parámetros varios. Debajo a la derecha, las opciones para visualizar la pieza, ya sea moviéndola linealmente, rotándola, agrandándola o disminuyéndola, o seleccionando directamente una de las vistas.

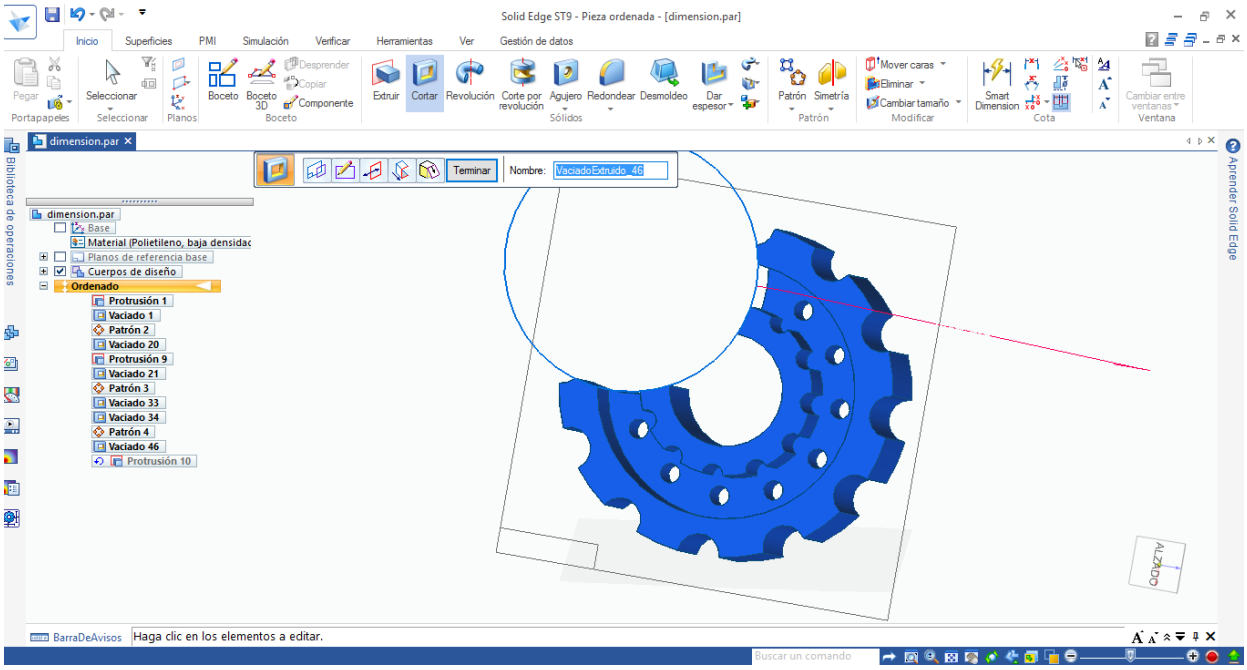

Figura 48: Imagen del botón casi acabada.

Siguiendo las operaciones una tras otra se llega a los últimos pasos antes de llegar a la pieza final. Como se puede observar, la pieza creada es bastante compleja, teniendo varias operaciones de vaciado con patrón, diferentes espesores e incluso un vaciado más grande donde se colocará la letra “D” para simbolizar la opción de dimensionar.

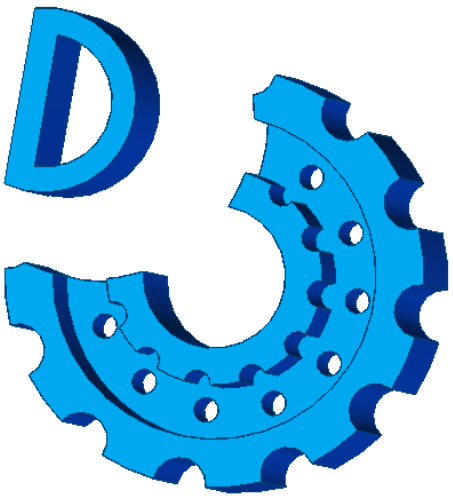

Figura 49: Imagen final para el botón de dimensionar.

Tras terminar de diseñar la pieza, retocarla y darla distinto color a las distintas superficies con Snagit Editor, se la vacía el fondo para su uso inmediato como imagen para el botón en la app.

A parte de las piezas más industriales, con Solid Edge también se puede crear diseños más artísticos.

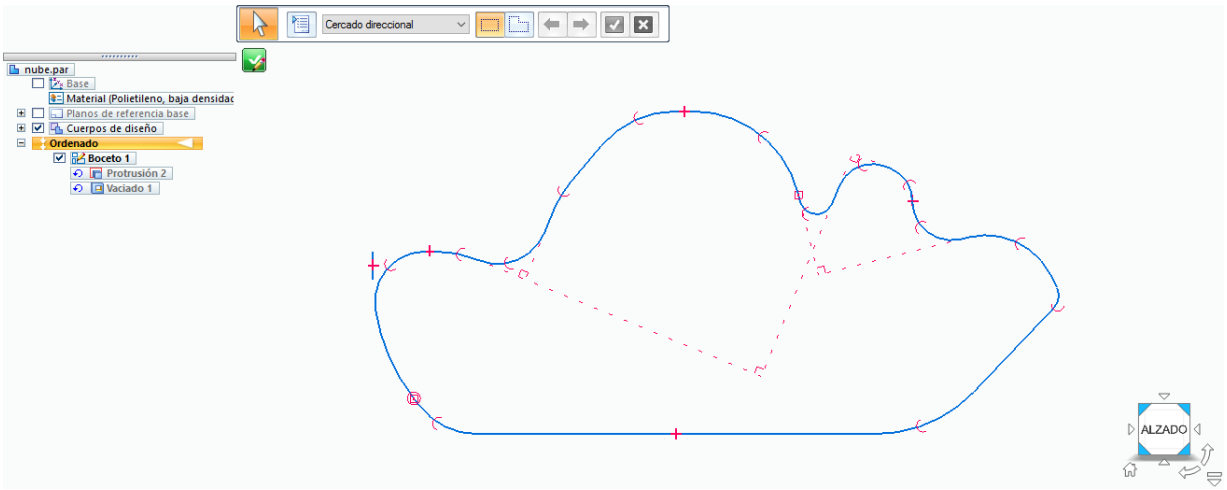

Figura 50: Boceto de la imagen del botón de descarga.

Llegando así a hacer la misma apariencia que una nube de verdad.

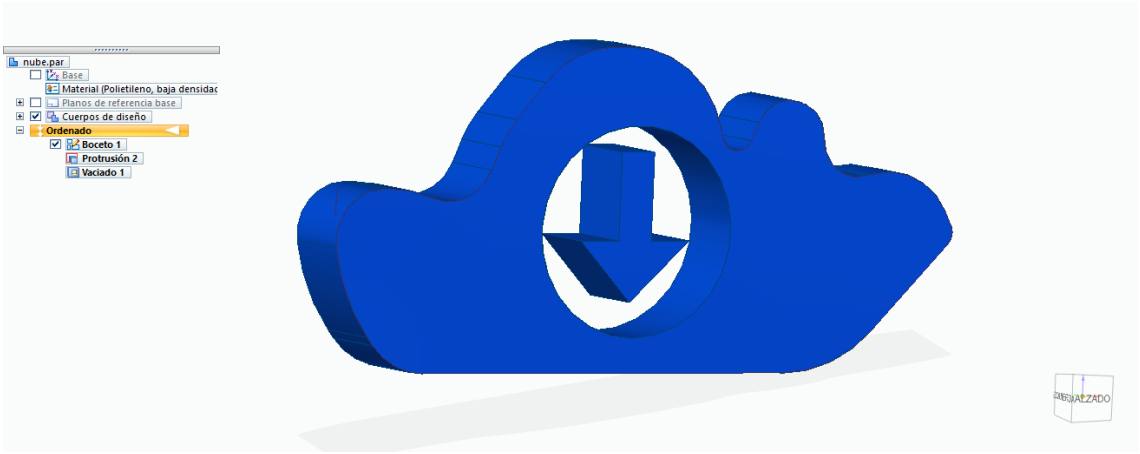

Figura 51: Modelo 3D de la imagen de descarga.

Del modelo final se observa como un dibujo 3D ha sido creado y cómo de vistoso puede llegar a ser.

Entre las múltiples opciones a las que tiene acceso el usuario, Solid le permite cambiar el material del dibujo. Esto sería más bien con vistas a las simulaciones, sin embargo, para la aplicación a desarrollar, permite poder conseguir tonalidades a la hora de pintarlo, a continuación se mostrarán distintos ejemplos de cómo usar los materiales disponibles.

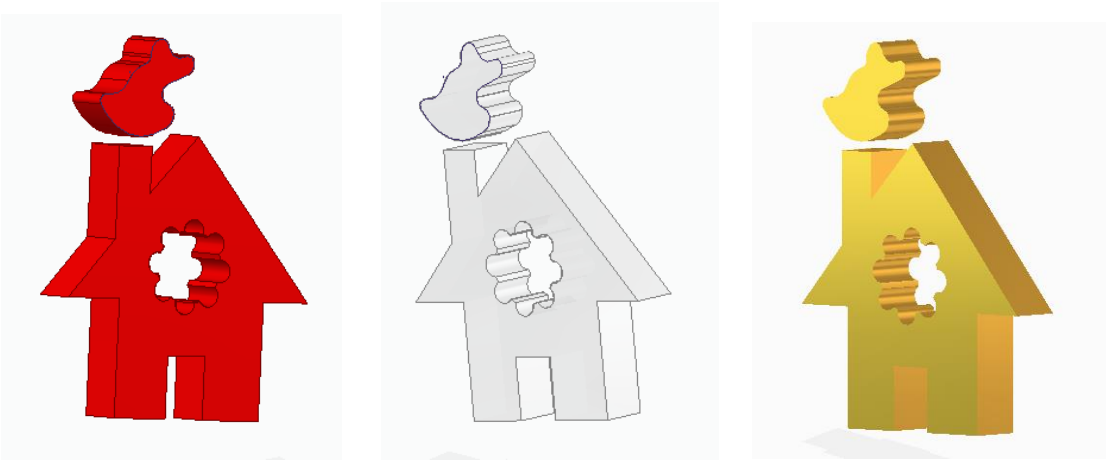

Figura 52: Distintos materiales en la misma figura.

De izquierdas a derechas tenemos ABS alto impacto, cristal industrial y oro. Los colores de los distintos materiales permiten una ran variedad de tonalidades ,que de tener que hacerlas a mano sería mucho más trabajoso, por lo que se le puede sacar un gran provecho a esta opción.

Cuando se explique el proceso de cómo se ha realizado la aplicación de calculo de volantes de inercia, se podrán apreciar los distintos iconos creados para cada comando.

4. METODOLOGÍA

4.1 Antecedentes

Este proyecto de crear una aplicación para el diseño de volantes de inercia, puede sonar a nuevo e innovador, pero alguien podría ya haber tenido esta o una ida parecida, haciendo que este proyecto no sea tan original, convirtiéndose así en meramente una copia.

Por ello antes de desarrollar la idea en el ordenador y empezar a escribir código, es importante saber si la idea que aquí se quiere exponer es nueva, o si alguien ya ha trabajado sobre ella antes. En caso de que haya ya una aplicación similar, se podrían tomar ideas para mejorar y distinguirse del resto dando más funciones, una mejor interfaz, más accesibilidad al usuario, etc.

Con objetivo de encontrar aplicaciones similares, se buscó en los dos principales sitios de aplicaciones que hay hoy en día en el mercado: App Store y Play Store (ambas a día 12/02/2017).

En la página de aplicaciones de Play Store, se encuentran aplicaciones con el nombre *flywheel* (volante de inercia) pero ninguna guarda relación con el proyecto en sí, más bien son juegos y aplicaciones de coches, nada que se pueda asemejar a la idea del proyecto. Tampoco se encontró buscando variantes con *volante de inercia.* Mirando ahora en App Store ya sí se encuentran dos aplicaciones que pueden competir con la aplicación a desarrollar.

La aplicación Flywheel Calculator, para IOS versión 5.0, creada el 02/10/2012, con precio de venta al público de 2,69 euros. La aplicación permite elegir entre tres tipos de volantes (hierro fundido, acero, otro material), dándole los radios, la velocidad y el ancho del volante calcula el peso, volumen, radio de giro y tensión en la cara del borde.

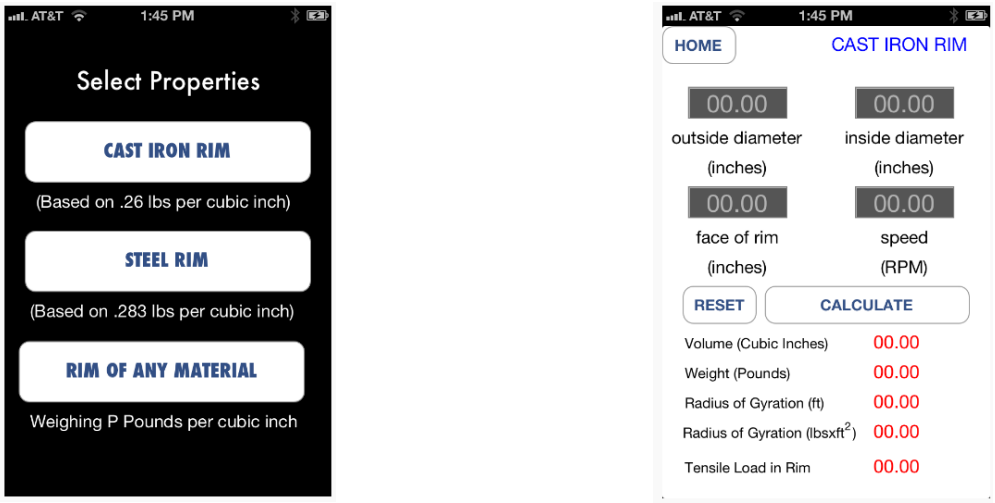

Figura 53: Visualización de la aplicación "Flywheel Calculator".

Siendo esto lo único que aporta la aplicación no es, ni de lejos; lo que en este proyecto se piensa ofrecer. Solo tres opciones a elegir y cuatro datos a meter no son suficientes como para calcular un volante de inercia en condiciones, por lo tanto; se debe seguir buscando posibles coincidencias, sabiendo ya que de momento no hay nada similar.

La otra aplicación FlyWheel Energy Storage Calc, para IOS versión 4.3, creada el 18/12/2010, con precio al público de 0,89 euros. La aplicación pide diferentes datos como los radios, la velocidad, el ancho del volante, la densidad, el tipo (no especifica cuales), potencia deseada y eficiencia. Después calcula el peso, volumen, omega, periodo, momento de inercia, energía almacenada y aceleración en el borde.

Figura 54: Visualización de la aplicación "FlyWheel Energy Storage Calc".

Esta ya sí se empieza a asemejar, no obstante no le da mucha operatividad ni opciones al usuario en cuanto a guardar datos, dar los puntos de la gráfica de la potencia, al igual que la ausencia de otros resultados. Es simple y fácil de usar, pero no es tampoco el nivel que se pretende dar a esta aplicación.

Zanjando así todas las aplicaciones para móvil en el mercado, se reafirma que la idea puesta en el proyecto es singular, hasta la fecha nadie ha creado una aplicación como la que se desarrolla en este proyecto. Siendo esto un aliciente a favor del proyecto no se puede quedar aquí la búsqueda, ahora habrá que mirar en otro tipo de plataformas como podrían ser las páginas web o programas para el ordenador. Esta última opción no influencia tanto en el

proyecto ya que la posibilidad de llevarlo a todas partes en el móvil no se la ofrece, sin embargo, coger ideas para la puesta en marcha de la app podría ser interesante.

Buscando en la red, no se encuentran grandes programas para calcular volantes de inercia, amén de catálogos de empresas que ofrecen sus productos, y que den algunas recomendaciones para su selección, no hay programa online que permita al usuario introducir los datos como para poder obtener unos resultados apropiados para la resolución de un volante de inercia. De entre las paginas buscadas, la más destacada sería Flywheel Energy Storage Calculator (de Calculator Edge).

Como ocurre con las aplicaciones mencionadas anteriormente, su maniobrabilidad es escasa; con solo tres datos a pedir no se es capaz de resolver un problema en condiciones. En la siguiente figura se muestra cómo es su interfaz y cuáles son los datos que se pueden introducir y qué resultados numéricos ofrece.

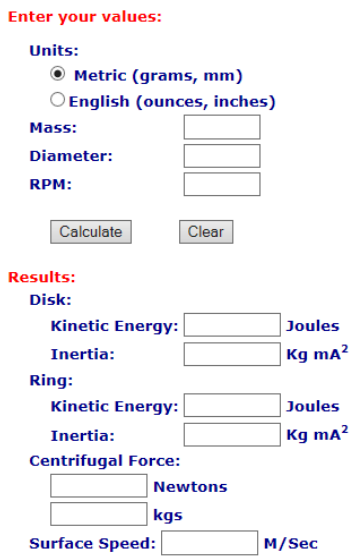

Figura 55: visualización de la web "Flywheel Energy Storage Calculator".

Como se venía diciendo, el programa no aporta la suficiente complejidad de resolución como para poder considerarse una opción válida y a tener en cuenta en la realización de este programa.

En lo que ha software para ordenador se puede encontrar, cabe destacar la referencia a un programa, YARCY 3.2 creado por alumnos de la Universidad Carlos III de Madrid (Leganés) para la asignatura Cinemática y Dinámica de Máquinas (Departamento de Ingeniería Mecánica). Este trabajo en grupo para la asignatura desarrollaba una aplicación informática, programada mediante MATLAB a través del entorno GUIDE.

Figura 56: Portada del programa YARCY 3.2.

YARCY 3.2 brinda al usuario la posibilidad de obtener los parámetros necesarios para el diseño de un volante de manera sencilla, con opciones tanto de visualización como de tratamiento de resultados que lo hacen especialmente atractivo para el usuario. Siendo esta la versión de programa que más se acerca al proyecto en cuestión, sirve como gran referente y fuente de inspiración para el presente Trabajo Fin de Grado.

El programa guía al usuario de manera fácil y llevadera por una serie de pasos hasta llegar a la resolución final del problema. Empezando por la distinción entre dos tipos de volantes, macizo y de brazos; siguiendo con la elección de par motor constante o para resistente constante, hasta la introducción de los datos del problema para su posterior resolución.

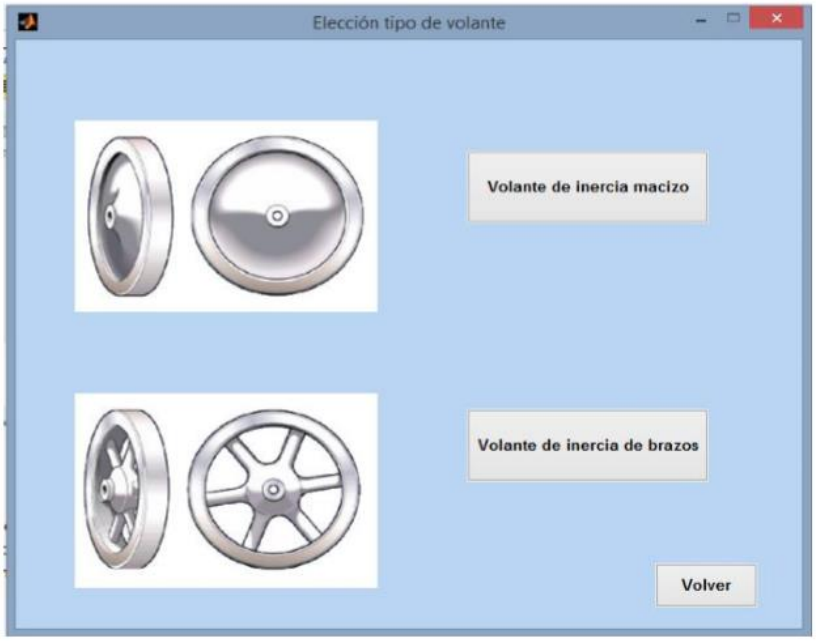

Figura 57: Selección del tipo de volante.

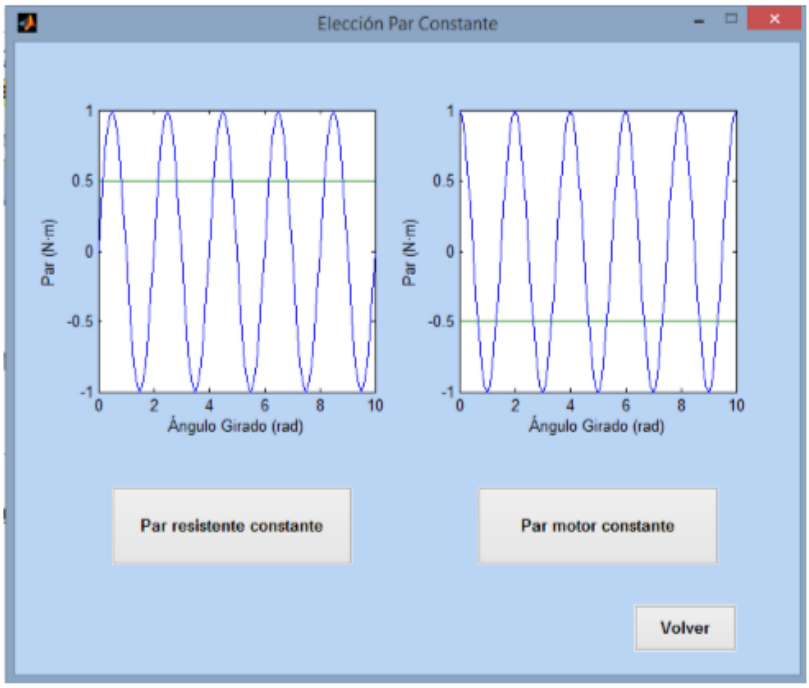

Figura 58: Selección entre par motor y resistente.

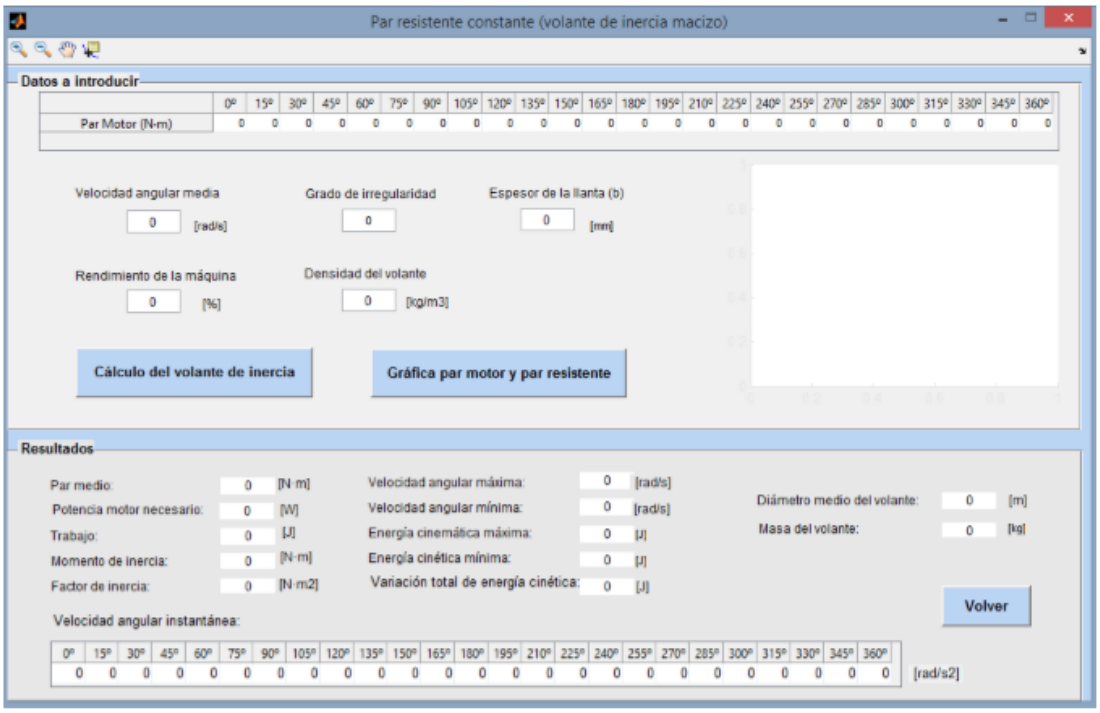

Figura 59: Pantalla donde meter los datos.

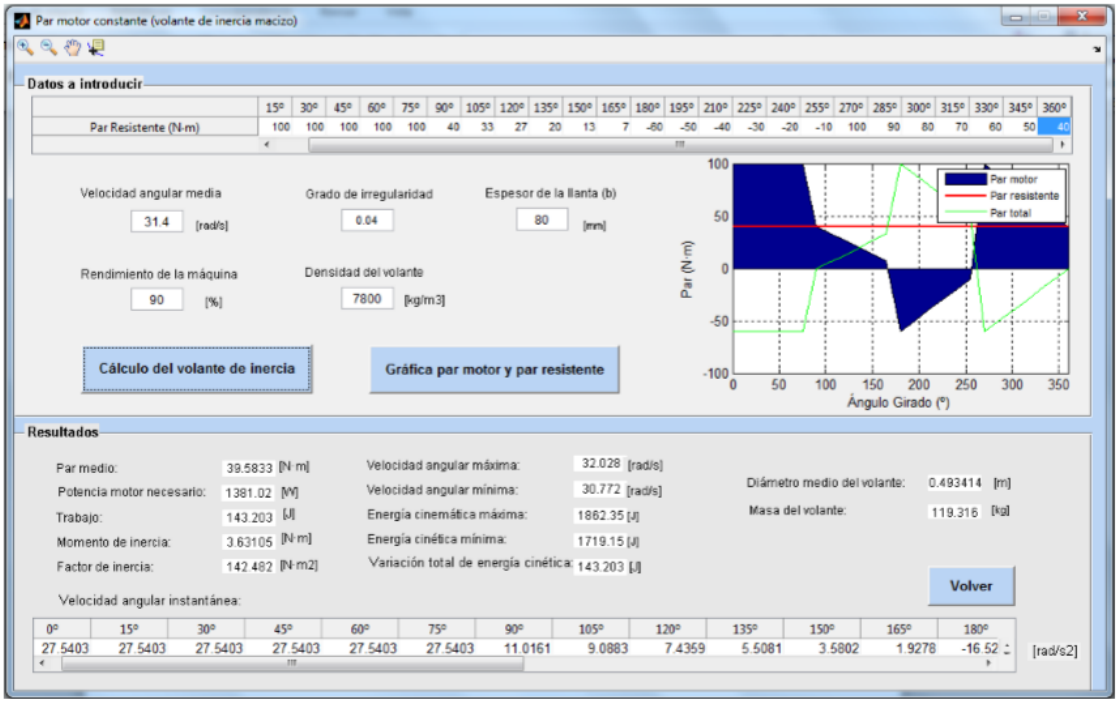

Figura 60: Ejemplo de un ejercicio resuelto en YARCY.

Los pasos a seguir son bien claros y simples, poniendo en cada hueco el tipo de dato que se requiere y en qué unidades hay que ponerlo. Un muy buen trabajo que allana el camino para el de este Trabajo de Fin de Carrera, al querer implementar a YARCY 3.2 en una mejorada versión para sistema Android, ofreciendo más opciones de manera sencilla y concisa como ya hicieron los compañeros con su trabajo.

4.2 Normativa sobre aplicaciones y derechos de autor

Las aplicaciones móvil son una gran ventaja para los usuarios, ayudando a satisfacer necesidades de todo tipo, dando entretenimiento por medio de juegos, posibilidad de comunicarse con otros usuarios por medio de las redes sociales, implementando características ya existentes del teléfono como la cámara o sensores, etc.

El beneficio que aportan es tal que muchas veces se pasa por alto el precio que tiene descargar una aplicación, no el monetario, ya que muchas aplicaciones son gratis o de bajo coste, si no al precio en la privacidad. Muchas de las aplicaciones que se descargan requieren de ciertas permisiones en datos personales del usuario, tales como nombre, fecha de nacimiento, número de teléfono, dirección de correo, fotos, contactos del teléfono, etc. Todos esto datos que a primera vista parecen de no gran relevancia, sí que suponen una amenaza para la privacidad ya que en algunos casos no se sabe para qué las van a utilizar o si quiera si las eliminarán una vez se prescinda de su servicio.

No es hasta hace un par de años cuando en 2013 las autoridades europeas de protección de datos, aprobaron el primer dictamen que clarifica el marco jurídico aplicable al uso de Apps para dispositivos inteligentes, (denominado Dictamen 2/2013 del Grupo de Trabajo del artículo 29 de la Directiva 95/46) concluyendo que resulta de plena aplicación la normativa de protección de datos (LOPD) para los desarrolladores de aplicaciones [31].

De entre las apps que se lanzan al mercado hay que diferenciarlas según su tipo de requisitos, siendo offline u online, dependiendo de si requieren internet una vez instaladas, e invasivas o no invasivas si una vez instalada la app se necesita de datos del teléfono como los ya descritos anteriormente.[30]

La aplicación para el cálculo de volantes de inercia será del tipo offline no invasiva. Una vez instalada no se requerirá ninguna otra conexión a la red puesto que toda la información necesaria para su uso y estará incluida en la misma aplicación. Al igual que los requisitos de instalación, no se necesita acceder a contactos del teléfono, ni las fotos ni a ningún otro dato personal del usuario, solo se requerirá un poco de espacio en la memoria interna del teléfono para poder guardar los datos de las simulaciones y calculo que el usuario haga y quiera mantener.

Al no tener necesidad de compartir información con el resto de usuarios, ni de tener anuncios publicitarios, al igual que por el tipo de app que se quiere hacer (utilidad técnica) no se vulnera ningún derecho de los usuarios menores en caso de que se lo descargasen. En el caso de las app y en España, se considera menor a los usuarios menores de 14 años de edad, en caso de requerir información personal tiene que avisarse a sus progenitores para su obtención.

Pese a cumplir con todos los requisitos y no tener ningún motivo por el cual poder sospechar de que se vaya a vulnerar la intimidad del usuario, en la página de donde se suba la app final, en el apartado de características se informara del tipo de aplicación que es (offline no invasiva) al igual de la necesidad de un espacio pequeño para la creación de los ficheros que guardan los archivos de las simulaciones que el usuario desee guardar.

Cumpliendo con las obligaciones que se adquieren al crear una app de cara a poner al público, solicitando el consentimiento especifico al usuario de un espacio en la memoria de su móvil para poder guardar los archivos que él crea convenientes , definiendo claramente la finalidad de la app e informando que una vez que prescinda de los servicios que brinda la app podrá desinstalarla sin problema alguno, se puede dar por bueno el cumplimiento de la normativa para aplicaciones móviles según la Directiva 95/46 de la LOPD.

Por ultimo añadir que las imágenes que en el programa salgan estarán referenciadas o serán propias, ya sean hechas a ordenador o tomadas con la cámara personal, además de las imágenes el código Java creado será propio, con todo esto se evitará entrar en derechos de propiedad intelectual. Al ser todo creado de cero sin calcar de ninguna otra fuente, se consigue tener una app única sin tener que dar cuentas a otros sitios.

Y en caso de que quisieran utilizar esta aplicación para usos comerciales o tomar las imágenes que en ellas se encontrase, la persona o entidad que lo hiciese estaría violando los derechos de propiedad de este Trabajo de Fin de Carrera si no pidiera permiso antes a su autor Ulises Martín Díaz.

4.3 Planning

Como ya se ha comentado en los respectivos apartados anteriores sobre las etapas del proyecto y la estructura del mismo, el esquema de las tareas y tiempos dedicados a ellas no es del todo lineal. Las diferentes actividades fueron llevadas a cabo en varios tramos, parte del escrito se fue haciendo a medida que se avanzaba en la recolección de datos, parte de la aplicación se hizo a medida que se encontraban las funciones en las diferentes referencias, etc.

No obstante, sí que se pueden marcar aproximadamente unos periodos de tiempo, ya que se concentran en ellos unas tareas más específicas. Con el siguiente diagrama GANT se refleja aproximadamente la sucesión de tareas con sus respectivos periodos temporales.

Tarea	Enero				Febrero				Marzo				Abril				Mayo				Junio	
Recopilación de datos teóricos de volantes.																						
Recopilación de referencias para la programación.																						
Pruebas de programación.																						
Creación de la app “Calculadora”.																						
Creación de la aplicación final.																						
Escrito del documento.																						
Implementación de la aplicación.																						

Tabla 4: Diagrama GANT

La franja temporal va desde Enero hasta mediados del mes de Junio. Los intervalos tomados son semanas, cuatro intervalos para los meses de Enero a Mayo, y solo dos para Junio. Dentro de cada intervalo (semana). También es de recalcar que durante el desarrollo de todas estas tareas y en sus múltiples intervalos no siempre fue la misma cantidad de horas semanales las que se las dedicó. En el apartado de costes se contabilizará el número de horas invertidas en cada parte, no obstante; la media de tiempo por semana para cada tarea ronda las 20 horas.

5. DESARROLLO DEL PROYECTO Y RESULTADOS

5.1 Concepción de la aplicación

Una vez vistos los antecedentes y cómo funciona Android Studio, es hora de encomendarse a la tarea de crear la verdadera app que permitirá al usuario calcular los volantes de inercia. Esto sería una ardua tarea si se quisiese abarcar todas las posibilidades, siendo estas casi infinitas (en cuanto a formas de pedir datos y de resolver en función de los mismos), se acotará el tipo de resolución a solo dos tipos de problemas.

Dando la posibilidad al usuario de insertar unos determinados valores y obtener un gran número de resultados, la aplicación se encamina en un cierto sendero y empieza a cobrar forma.

Antes de embarcarse en la disertación de cómo se llega a los distintos resultados, se establecerá el orden secuencial que seguirá la aplicación. Mediante estos diagramas de bloques se muestra las distintas pantallas.

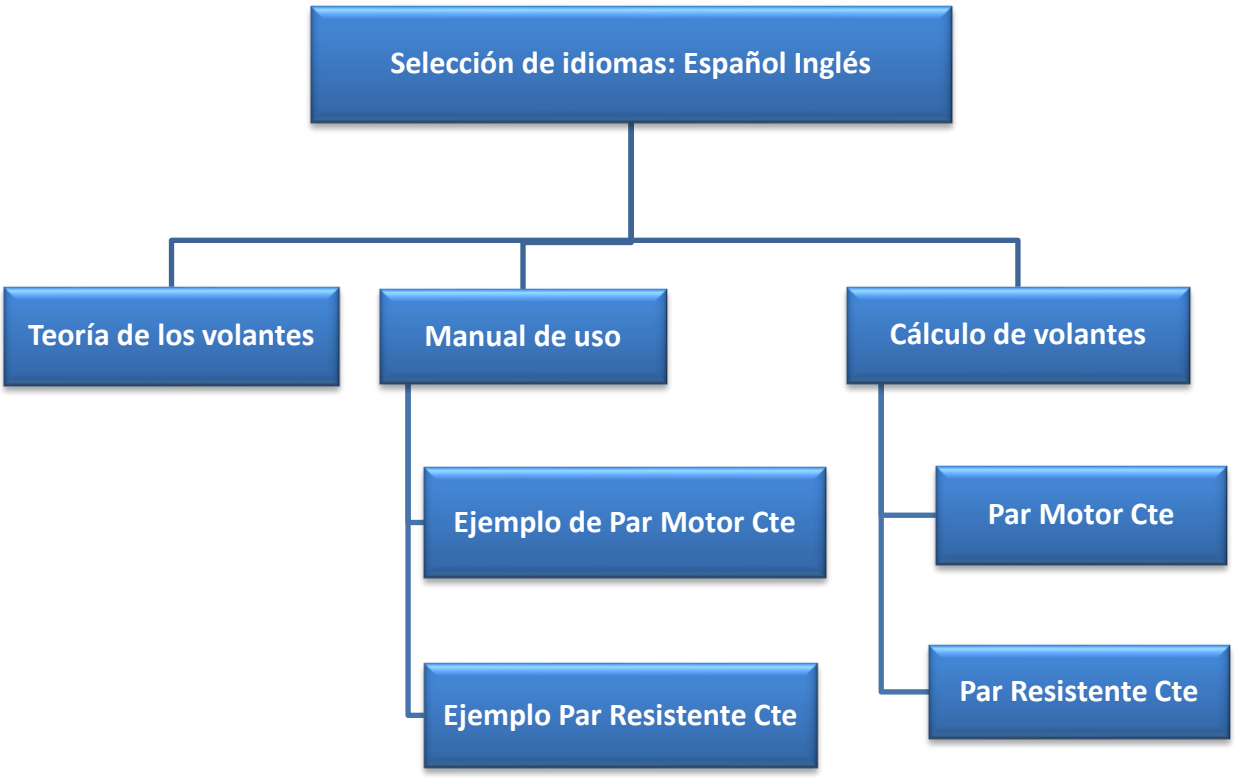

Figura 61: Diagrama de flujo de la aplicación.

Figura 62: Diagrama de flujo de la aplicación (continuación).

En estos dos gráficos de jerarquía e muestra como está estructurada la aplicación en sí. Partiendo de la "Portada" se elegiría el idioma en el que se quiere la app. Una vez elegido estaría el "Menú" donde estarían las opciones de ver la "Teoría", "Manual de uso" con los ejemplos y "Calcular" los volantes. Todas las actividades (pantallas) tienen la opción de "Volver" retrocediendo a la actividad anterior.

Dentro de la pantalla "Calcular" se seleccionaría el tipo de problema a resolver, ya sea "Par Motor Constante" o "Par Resistente Constante". Indiferentemente de cuál se elija, los datos a rellenar será, los mismo; teniendo que aportar una serie de datos y seleccionando el "Número de Vueltas" (1/2) y el "Número de Divisiones" (6/12/24) para la gráfica del par.

Con ambos parámetros el programa dará al usuario permiso para rellenar los puntos correspondientes de la gráfica según haya seleccionado previamente. A continuación, al resolver se mostrará la gráfica creada junto con los resultados del problema. Por si fuese muy pequeña la imagen de la gráfica, se facilita la opción de "Ampliar Gráfica", apaisando la pantalla y aumentando el tamaño del recuadro del visor.

Si por el contrario se quiere dar dimensiones al volante calculado, la opción "Dimensionar" permitirá pasar a una pantalla donde se especificará unos datos para obtener las medidas del volante resuelto.

En todos los apartados donde se introducen datos se da la posibilidad de "Guardar" y "Cargar" permitiendo así tener unos datos ya metidos por si se quiere repetir la simulación anterior o solo modificar parte de ella.

Para tener un mejor sentido del orden y de cómo iría estructurada, se recomienda ir al apartado de ANEXOS y ver el sub-apartado de Actividades de la aplicación "Cálculo de Volantes de Inercia", donde se muestra en diferentes recuadros simulando el teléfono, qué habría en cada pantalla.

5.2 Selección del tipo de problemas

Con la aplicación ya en mente, se puede acotar con mayor facilidad cómo serán los problemas que el usuario podrá resolver en ella. Partiendo del objetivo de que se tiene en conocimiento ciertos datos acerca de la máquina a la cual se le quiere acoplar el volante, los datos a suministrar por el usuario serían [15][17][4]:

- Irregularidad δ [%]
- Velocidad de Régimen ω_m [rpm] (Como ya se comentó anteriormente, se aproxima la velocidad de régimen a la velocidad media por su escasa diferencia)
- Rendimiento de la Máquina η [%]
- Curva del Par Motor o Resistente [N m] (Dependiendo del ejercicio que se quiera resolver). Seleccionando entre vueltas y divisiones:
    - Número de Vueltas 1 (360^o) 2 (720^o)
    - Número de Divisiones (6/12/24)

Con estos datos el usuario obtendrá los siguientes resultados:

- Par Medio (M_m) [N m]
- Potencia Necesaria (P) [W]
- Momento de Inercia (I_V) [kg m^2]
- Factor de Inercia (PD^2) [N m^2]

- Velocidad Angular Max (ω_{max}) [rad/s]
- Velocidad Angular Min (ω_{min}) [rad/s]
- Trabajo (T) [J]
- Energía Cinética Max (E_{max}) [J]
- Energía Cinética Min (E_{min}) [J]

5.3 Obtención de los resultados

Partiendo de las hipótesis previamente mencionadas y de los conocimientos teóricos descritos en los apartados anteriores, los diferentes resultados se obtendrían de la siguiente manera:

Par Medio (M_m)

Con los datos seleccionados del número de vueltas y número de divisiones, el usuario podrá rellenar con los distintos valores que crea correspondientes, cada una de las entradas para cada punto. El programa hallará el área encerrada entre la gráfica y el origen y dividirá por el número de divisiones elegidas para encontrar $\mathbf{M_m}$.

Potencia Necesaria (P)

La potencia que requerirá el motor según se haya fijado su rendimiento **η**, teniendo en cuenta la velocidad de régimen $\boldsymbol{\omega_m}$ y el dato calculado anteriormente $\mathbf{M_m}$, se calcula con la siguiente fórmula:

$$P = \frac{\mathrm{M_m}\ \omega_m}{\eta}$$

Momento de Inercia (I_V)

El momento de inercia del volante se calcula con los parámetros ya conocidos de la velocidad de régimen $\boldsymbol{\omega_m}$ y de la irregularidad $\boldsymbol{\delta}$, además del área **A** que será el nuevo

trabajo, dicho dato saldrá del sumatorio de las áreas positivas o negativas (ya que darán el mismo resultado numérico) de la intersección del par medio $\mathbf{M_m}$ con el diagrama. Quedando así la siguiente expresión:

$$I_V = \frac{A}{\delta\ \omega_m^2}$$

Factor de Inercia (PD^2)

El factor de inercia, que es el verdadero valor que se da para conocer el volante de inercia, vendría dado por la expresión:

$$PD^2 = \frac{4\ g\ A}{\delta\ \omega_m^2}$$

Velocidades Angulares ($\omega_{max}\ \omega_{min}$)

Para el cálculo de ambas velocidades extremas, se usará la irregularidad $\boldsymbol{\delta}$ fijada y la velocidad de régimen $\boldsymbol{\omega_m}$, obteniendo:

$$\omega_{max} = \omega_m \left(1 + \frac{\delta}{2}\right)$$

$$\omega_{min} = \omega_m \left(1 - \frac{\delta}{2}\right)$$

Energías Cinéticas ($E_{max}\ E_{min}$)

Para ambas energías la fórmula será la misma, solo cambiando la velocidad angular.

$$E_{max} = \frac{1}{2}\ I_V\ \omega_{max}^2$$

$$E_{min} = \frac{1}{2}\ I_V\ \omega_{min}^2$$

Trabajo (T)

Finalmente, el trabajo se obtendrá por la resta de la energía cinética máxima $\boldsymbol{E_{max}}$ menos la mínima $\boldsymbol{E_{min}}$.

$$T = E_{max} - E_{min}$$

5.4 Creación y generación del código de la app

Como ya se explicó en el apartado de descripción de las herramientas utilizadas, el programa el cual se usará para desarrollar la aplicación será Android Studio. Partiendo de un nuevo proyecto en blanco, como se empezó la app de la calculadora; se empieza a dar forma a las diferentes actividades [6][24][25].

Siguiendo un razonamiento estructurado, y bajo las pautas del diagrama jerárquico anterior, se empieza por la versión en español, ya que la versión en inglés será una mera copia y traducción del mismo.

La primera actividad (pantalla) que se tendrá que crear, siguiendo lo descrito anteriormente; es la portada donde se incluirá el nombre de la aplicación junto con el logo de la misma y ambas opciones en las que elegir el idioma. Como la primera imagen que se va a ver tiene que ser muy vistosa y creativa, se opta por el diseño de un logo que resemble un volante de inercia. Este se hará mediante el uso del programa Solid Edge para el diseño puro y el Snagit Editor para su posterior pintado y acabado.

A medida que se crea este logo, se observa que no solo el logo estará a la vista, sino que el fondo de pantalla jugará un papel importante, ya que realzará la imagen y dará una sensación de cohesión al usuario.

Tras varias pruebas de colores y formas se opta por la siguiente configuración de la pantalla de inicio.

Figura 63: Portada de la app.

La pantalla de inicio da una impresión agradable a la vista, el paso de azul más oscuro a más clarito con tantos intervalos, hace que no sea muy agresivo, atenuando así el cambio de un color a otro. La imagen principal es el rótulo del nombre de la aplicación “CaVI”

Viendo ahora la parte del código que hace que esto se sostenga, se han de mirar dos archivos, el primero referente a la colocación de los botones imágenes y texto; y la segunda referente a la asignación de las acciones correspondientes a esos botones.

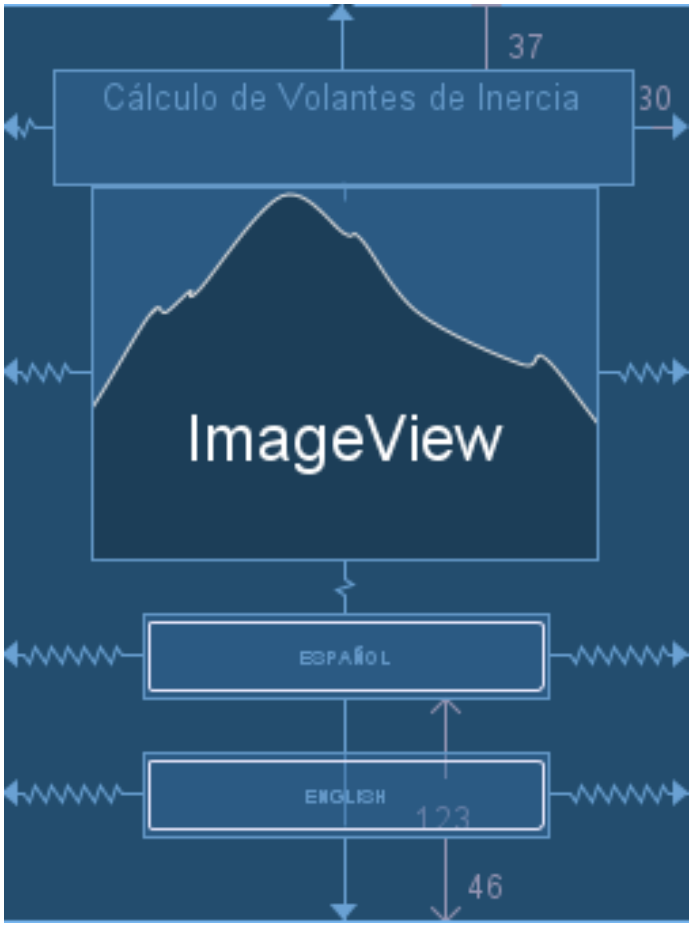

Figura 64: Colocación de los objetos en la portada.

Esta sería la vista de cómo está colocado cada elemento, fijándolo a cada lado (componente X e Y) para que al abrir la aplicación no se desordenen. Si se fuese a la vista de propiedades del código, se podría apreciar los otros factores como la asignación del fondo mediante la carpeta "*drawables",* la cual contiene todas las imágenes externas que se quieran usar en la aplicación.

Cambiando al archivo de las funciones a utilizar, para este caso en concreto; no se usan muchas, ya que solo tenemos dos botones, los cuales al ser pulsados nos tendrían que llevar respectivamente a la nueva actividad de "*menú"* ya fuese en español o en inglés.

Durante el desarrollo de las diferentes actividades, se observa el efecto del fondo de pantalla utilizado en la portada sobre los diferentes textos y recuadros y no acaba de encajar. El tono azul, pese a ser clarito; no deja apreciar bien del todo los números y letras. Haciendo dificultosa o más tediosa el uso de la aplicación, se opta por unos tonos aún más claritos, haciendo que los textos se realcen más.

La siguiente pantalla que plantea una dificultad añadida es la del "*cálculo de volantes de inercia",* saltando las precedentes actividades de "*teoría" y "manual de uso"* ya que no hay mucho más que profundizar en ellas. Su diseño es parecido al de la portada, ambos casos constan de una serie de imágenes en una pantalla deslizante, la cual mostrará la teoría de los volantes de inercia o los ejemplos de cómo usar la aplicación. No obstante, ambas opciones se pueden visualizar en el apartado de anexos.

Tomando como ejemplo la pantalla de "*Par Motor constante*", se muestra el cambio de fondo de pantalla y los diferentes elementos nuevos que se añaden para la creación de la misma.

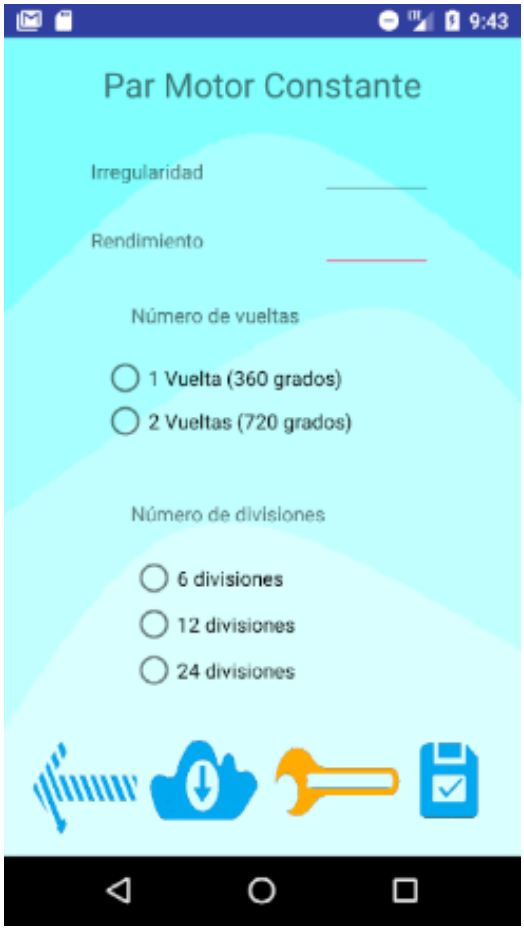

Figura 65: Par motor constante.

Empezando por lo visual. El fondo de pantalla, siguiendo con el estilo de la portada en forma de olas; se aclara realzando el contenido que en ella se encuentra. Los tonos azules claritos, aún más claritos que los anteriores; mejoran la visualización de lo que se pide y no

hacen que se esfuerce tanto la vista. Los iconos que aparecen también son propios, todos creados y retocados con los programas Solid Edge y Snagit Editor.

Cada icono es un botón que ejemplifica con la imagen lo que viene a hacer, siendo la flecha la que indica ir para atrás, la nube la opción de descargar, la llave la opción de resolver y la del disquete la de guardar. También se ha querido denotar con otro color distinto al del resto, la opción que continuaría; creando un objeto más visual para que por instinto el usuario pulse ese botón y continúe sin desviarse del camino principal.

Pasando al plano del código, se aprecia que se han añadido varios tipos de comando distintos, entre ellos la opción de poner un visor deslizante, botones "radiobutton", textos interactivos, opción de guardado y de carga de datos.

La opción del visor deslizante (scroll view), se utiliza en los casos que la pantalla del aparato sea más pequeña que el del contenido que se quiere poner, como ya se verá más adelante para las opciones de meter puntos, el espacio deja de ser un problema con esta opción. Los comandos para poner este tipo de visualización no son más distintos a los normales (vertical layout, horizontal layout, los cuales crean un recuadro en el que meter contenido, fijándolo en su respectiva orientación).

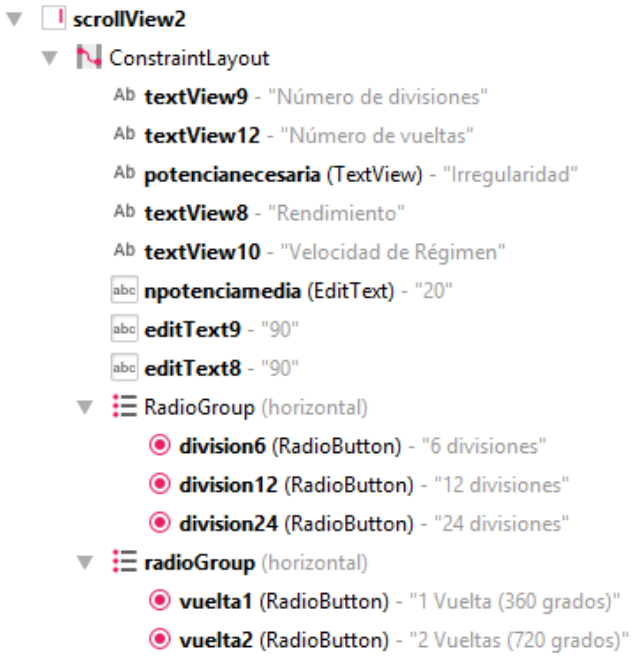

Figura 66: Contenido de la ScrollView.

Este es el contenido que tiene la scroll view, al ser muy extenso no entraría en la pantalla del teléfono, independientemente del layout puesto, por lo que es muy útil para poner varias opciones o textos o imágenes muy amplias.

Otro de los comandos nuevos que se utilizaron es el RadioGroup y los RadioButton, los cuales le dan la opción de elegir entre varias opciones al usuario pero solo activa una por cada RadioGroup. Esta opción permite, sin complicar mucho el código; dar tantas opciones se quiera sin la preocupación de que vaya a marcar más de una pudiendo inducir al fallo de la aplicación.

Para mayor seguridad, se creó un comprobador para que al menos se hubiesen marcado una opción de cada RadioGroup, y en caso de no tener una de cada marcado saltase un mensaje para pedir que se marcasen, impidiendo también que se continúe a otras actividades (ya que al no estar marcadas el programa no sabría a cuál ir y se detendría).

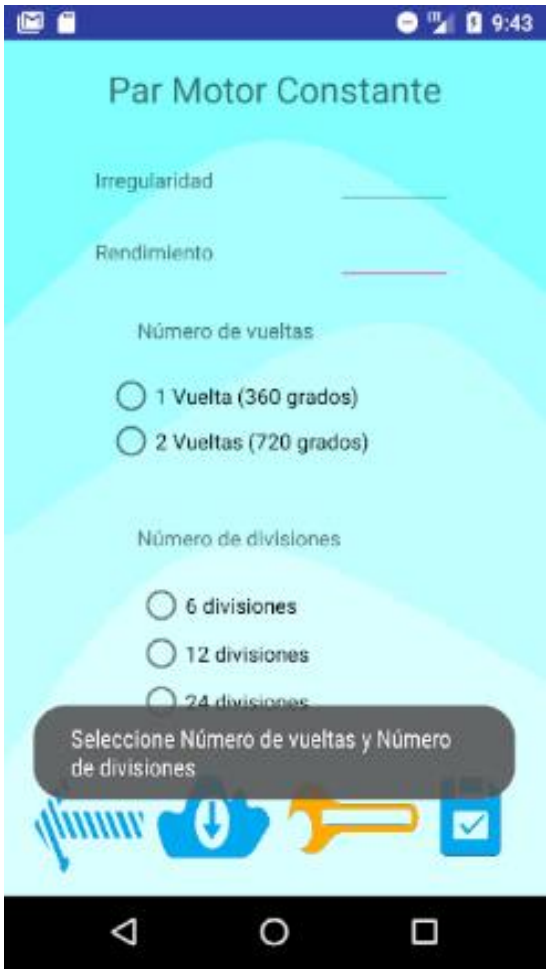

Figura 67: Aviso de seguridad.

Esto asegura que incluso si el usuario toca la tecla de resolver sin querer, aun no habiendo seleccionado el número de vueltas ni el número de divisiones; el programa no se detendrá, solo aparecerá un cartel avisando de que faltan esos parámetros por contestar. Esto se incluirá dentro de la sección de errores encontrados y futuras mejoras.

El usuario tiene que poder meter de alguna manera los datos de la irregularidad, velocidad de régimen y el rendimiento del motor. Todos estos parámetros son números enteros positivos, ya dentro del programa se tratarán de la manera debida. Por lo tanto, dentro de las características de estos "EditText" (manera de permitir que se modifique un texto por el usuario) se pondrán las limitaciones de solo usar números positivos enteros. Con estas limitaciones se puede asegurar que no se introduzca ningún número negativo.

Por último las opciones de *"guardar"* y *"cargar"*. Ambas van relacionadas, ya que una vez que generas el código para reservar un espacio dentro del teléfono, solo tienes que llamarlo para que se pueda leer y meter en su debido lugar.

La opción de guardar para esta actividad permitirá al usuario guardar los valores de la irregularidad, velocidad de régimen y rendimiento del motor. De esta forma si quiere podrá repetir los ensayos con esos mismos números cambiando o no el número de vueltas y divisiones. Para cargar solo hay que llamar a la memoria interna para que el programa pueda leer los archivos y colocarlos en sus respectivas posiciones; en caso de no haber un valor previo, no habría problema alguno ya que en caso de no haber valor precedente guardado no pasaría nada, el botón estaría desactivado.

```
public void guardar (View view)
{
    String number1=value0A.getText().toString();

    String file1="helo_file1";
    try {
        FileOutputStream fileOutputStream = openFileOutput(file1,MODE_PRIVATE);
        fileOutputStream.write(number1.getBytes());

        fileOutputStream.close();
        Toast.makeText(getApplicationContext(),"Dato guardado",Toast.LENGTH_LONG).show();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    String number2=value1A.getText().toString();

    String file2="helo_file2";
    try {
        FileOutputStream fileOutputStream = openFileOutput(file2,MODE_PRIVATE);
        fileOutputStream.write(number2.getBytes());

        fileOutputStream.close();
        Toast.makeText(getApplicationContext(),"Dato guardado",Toast.LENGTH_LONG).show();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

Figura 68: Código para guardar.

Parte del código que hace que esta opción funcione sería la imagen anterior, mediante una serie de permisos y comandos dejamos que la aplicación meta una información dentro de la memoria interna del aparato móvil, esta información se ha codificado para que sea privada, es decir; otras aplicaciones no pueden acceder a ella, solo esta. Los datos metidos son el nombre del archivo por el cual se va a reconocer su llamada y el valor del campo seleccionado.

A la hora de cargar la información lo que se hace es llamar dentro de la memoria interna al archivo que se ha guardado previamente por el nombre que se le asignó. Este nombre es fijo por lo que si se guardan más de una vez, el valor grabado será siempre el último, sin posibilidad de poder salvar o recordar veces anteriores.

```
try {
    String number3;

    FileInputStream fileInputStream = openFileInput("helo_file3");
    InputStreamReader inputStreamReader= new InputStreamReader(fileInputStream);
    BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
    StringBuffer stringBuffer3 = new StringBuffer();


    while ((number3=bufferedReader.readLine())!=null)
    {
        stringBuffer3.append(number3);
    }
    value2A.setText(stringBuffer3.toString());

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
Toast.makeText(getApplicationContext(),"Datos cargados",Toast.LENGTH_LONG).show();
```

Figura 69: Código para cargar.

Para asegurar al usuario de que ha guardado o cargado correctamente los datos, saldrá un mensaje diciendo “Dato guardado” o “Dato cargado”, como se hizo para la opción de asegurar que se hubiese seleccionado los parámetros de vueltas y divisiones antes de resolver.

Una vez se han rellenado los apartados correspondientes y seleccionado los parámetros para meter los datos de la gráfica (par motor o resistente), al pulsar resolver se abrirá la actividad de la gráfica.

En caso de que se hubiese seleccionado dos vueltas y doce divisiones, la apariencia de la actividad sería la siguiente.

Figura 70: Pantalla donde meter los puntos.

Como se muestra en las imágenes, cada 60º se tiene un punto hasta alcanzar las dos vueltas (720º). En cada entrada se le permite al usuario meter un valor para cada punto. Como pasaba en la actividad anterior, hay que limitar el tipo de input que se le va a permitir meter al usuario. Para este caso se requiere la posibilidad de meter números enteros tanto positivos como negativos, ya que se quiere meter la solicitación del par resistente o del par motor.

Para esta actividad se ha habilitado la posibilidad de guardar y cargar como en la anterior pantalla. Aquí lo que se guarda y posteriormente se carga, son los valores asignados a cada punto. A diferencia de la actividad anterior, para este caso se ha permitido grabar en cada combinación de vueltas y divisiones; teniendo un número total de posibilidades de guardado de 6 opciones, ya que se comparten las opciones para par motor y par resistente dentro de la misma opción.

Una vez seleccionados todos los valores de los puntos se procederá a pulsar la opción de resolver, la cual abrirá la actividad de *"Soluciones"* para que el programa calcule todos los datos referentes a las soluciones y muestre la gráfica que se ha generado.

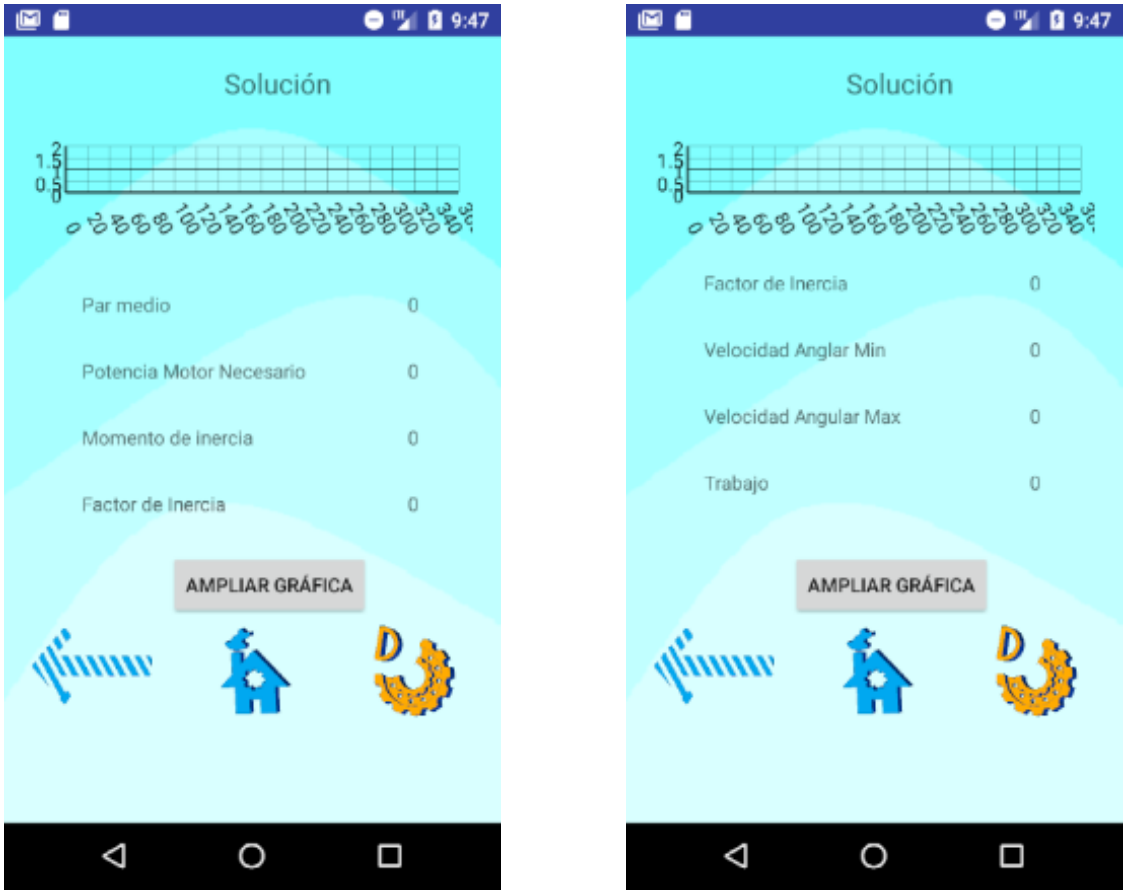

Figura 71: Pantalla de los resultados.

Después de haber pasado por la recolección de datos de las actividades anteriores, finalmente se pueden obtener todos los resultados que se mencionaron en el comienzo. De todos las variables que se necesitan para calcular los distintos parámetros de la solución, la más importante es "A", el trabajo de la gráfica. Una vez se tiene A es cuestión de meter en las ecuaciones del principio y los resultados saldrán sin ninguna complicación.

Lo que a primera vista se podría percibir como algo relativamente sencillo de hacer, en realidad no lo es ya que en papel se ve y se sabe dónde está cada cosa, ya que se tienen nociones de dónde mirar y que es positivo y que negativo. Pero el programa solo lee códigos, códigos que tienen puntos y coordenadas, nada más.

La obtención en primer lugar del *"Par medio"* se hará mediante un proceso de iteración, cogiendo cada dos puntos y obteniendo la recta que se genera entre ellos. Con los valores de la pendiente "m" y el punto de intercepción en la ordenada "n" se podrá integrar entre los dos puntos de manera que se obtenga el valor del área encerrada.

Punto 0	X_0	Y_0
Punto 1	X_1	Y_1

Tabla 5: Tabla de puntos.

$$n_0 = y_0 \qquad m_0 = \frac{y_1 - n_0}{x_1}$$

$$y_0 = m_o\, x_0 + n_0$$

$$A_0 = \frac{m_0\, x_1^2}{2} + n_0 x_1$$

Repitiendo el proceso hasta llegar al último punto, y sumando en cada tramo el valor del área encerrada, se obtiene el valor total del área de la gráfica. Después se dividirá por el número de intervalos obteniendo el valor del *"Par medio".*

No obstante, falta por calcular el valor "A" (trabajo) que es el que permite calcular gran parte de los valores. Para ello se necesitará otro proceso iterativo, en este caso, aún más complejo ya que se cambia el eje de corte.

Tras varios intentos, se llega a la conclusión que el cambio de eje no es más que un cambio en la coordenada "n" de cada recta, sería como bajar la gráfica una distancia igual al par medio. Sin embargo, falta hacer una discriminación de los puntos de cortes, ya que cada vez que se corte el nuevo eje, cambiará de signo el área encerrada, y como se quiere tener a un lado la suma de todas las áreas negativas y a otro las positivas (ambos sumatorios tendrá un valor final igual) es necesario modificar de nuevo el código.

Para el caso anterior los intervalos siempre valían lo mismo, ya que todos eran equidistantes, sin embargo; si ahora la recta corta el nuevo eje el intervalo se divide en dos, haciendo que la fórmula anterior no sirva tal cual está. Para empezar a calcular las nuevas áreas se crea un bucle para cada tramo, verificando si corta o no corta, en caso de que corte se integra hasta el punto de corte y se suma ese valor a una variable A(b). El resto del tramo se integra desde el punto de corte hasta el final, sumando ese área a otra variable A(b+1), si no cortase todo se integraría desde principio a fin sumando el área a la variable A(b) como se venía haciendo en el caso anterior.

Así se procedería para cada intervalo hasta el último, cada vez que se cortase se cambiaría de variable incrementándola por uno (b+1), de esta manera, al final de la iteración tendremos n+1 tramos de área, es decir n+1 variables de A donde n es el número de veces que se ha cortado el nuevo eje de coordenadas.

Tras eso se comprobaría cada área, si es positiva se suma a una nueva variable A_{Total} la cual vendrá a ser el trabajo que se quiere hallar. También se podría hacer con las áreas negativas ya que el valor final sería el mismo.

Para hacer esto, se modifica las ecuaciones anteriores substrayendo el valor *"Par medio"* (M_m) a el punto de intercepción en la ordenada "n". Con esto se bajaría la gráfica como se ha explicado anteriormente. Una vez esto se comprueba si corta o no el tramo y dependiendo de eso se integra completa o a tramos.

Punto 0	X_0	Y_0
Punto 1	X_1	Y_1

Tabla 6: Tabla de puntos.

$$n_0' = y_0 - M_m \qquad m_0 = \frac{y_1 - n_0'}{x_1}$$

$$y_0 = m_o\, x_0 + n_0'$$

$$cx_0 = \frac{M_m - n_0}{m_1}$$

If $cx_0 > 0$ "corta"

$$A(b) = \frac{m_0\, cx_0^2}{2} + n_0' cx_0$$

b = b + 1

$$A(b) = \frac{m_0\, x_1^2}{2} + n_0' x_1 - \left(\frac{m_0\, c x_0^2}{2} + n_0' c x_0\right)$$

Else “no corta”

$$A(b) = \frac{m_0\, x_1^2}{2} + n_0' x_1$$

Una vez se hayan hecho todos los tramos se comprueba si es positivo o negativo el área y se irían sumando, llegando así al valor A que es el trabajo que se busca. Con este valor ya se pueden calcular todos los resultados listados anteriormente.

Como bien se dijo en la descripción del programa, Android Studio da una muy amplia selección de funciones y opciones a la hora de programar la aplicación, no obstante; para requisitos fuera de lo común se necesita importar otras librerías. Esta opción de implementar la librería de Android Studio con otras sacadas de su misma web o creadas por otros usuarios, abre un sinfín de nuevas oportunidades. Para este proyecto se requisó de una librería para la confección de gráficas, puesto que uno de los detalles que ofrece es la visualización de los puntos que el usuario da para representar el par motor o par resistente.

Dicha librería es proporcionada por el mismo Android Studio bajo la categoría de Android Graph View [23]. En esta página web se puede descargar la nueva librería y añadirla a la que está siendo usada para el proyecto en cuestión. Además de dar la posibilidad de usarla, incluye tutoriales y explicaciones de cómo usarla y cómo sacarle provecho a esta herramienta.

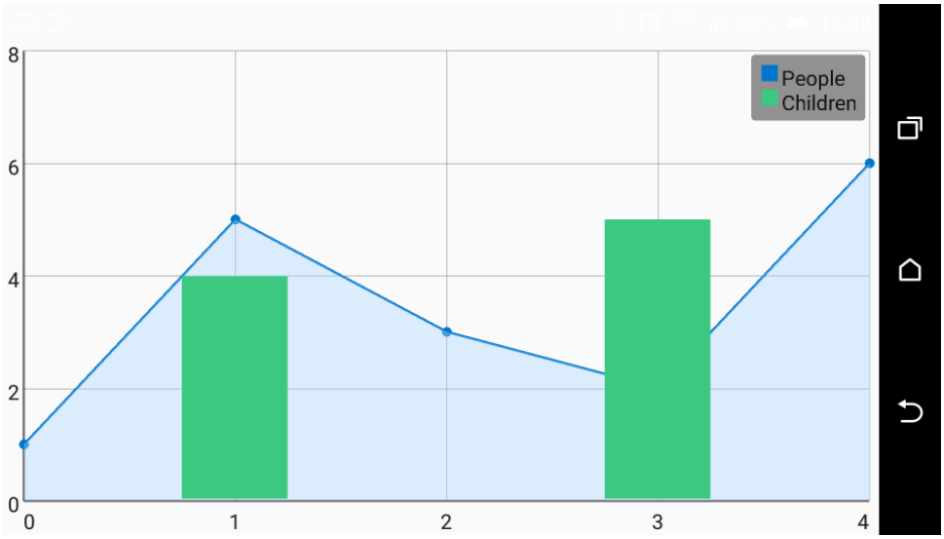

Figura 72: Visualización de la gráfica por Graph View.

Al ser una extensión que el mismo Android proporciona no hay que preocuparse por la compatibilidad entre la librería del proyecto y la incorporación de la nueva. Es una herramienta fácil de usar que ayuda a la creación y visualización de todo tipo de gráficas, ya fuesen de puntos, líneas, barras, circulares, etc. Esto incluye incluso, la posibilidad entre combinarlas entre ellas y de usarla junto a otros dispositivos del móvil, como por ejemplo el pulsímetro, velocímetro, etc.

Figura 73: Gráfica acoplada a un velocímetro.

En este caso se utilizará para la visualización de los puntos que el usuario haya dado para representar la gráfica par motor o resistente. Descargando le archivo de la web, se instala directamente en el fichero de la librería del proyecto. Tras acoplarse con éxito se procede a la utilización de la nueva herramienta.

```
<com.jjoe64.graphview.GraphView
    android:id="@+id/graph"

    android:layout_width="match_parent"
    android:layout_height="177dp"
    android:layout_marginBottom="10dp"
    android:layout_marginEnd="20dp"
    android:layout_marginLeft="25dp"
    android:layout_marginRight="20dp"
    android:layout_marginStart="25dp"
    android:layout_marginTop="20dp"
    android:visibility="visible"
    app:layout_constraintHorizontal_bias="0.473"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Figura 74: Código de la visualización de la gráfica.

Este es el código que hay que escribir para poder usar la opción de visualizar la gráfica en la pantalla. Se le podría añadir más líneas para incrementar las opciones como por ejemplo, añadir un sombreado, cambiar el color de las líneas, marcar máximos o mínimos, etc.

```
    nb0 = Integer.valueOf((n0));
    nb1 = Integer.valueOf((n1));
    nb2 = Integer.valueOf((n2));
    nb3 = Integer.valueOf((n3));
    nb4 = Integer.valueOf((n4));
    nb5 = Integer.valueOf((n5));
    nb6 = Integer.valueOf((n6));

    GraphView graphView = (GraphView) findViewById(R.id.graph);
    LineGraphSeries<DataPoint> series = new LineGraphSeries<>(getDataPoint());
    graphView.addSeries(series);
    series.setDrawBackground(true);
}

private DataPoint[] getDataPoint() {


    DataPoint[] dp = new DataPoint[]{

            new DataPoint(0, nb0),
            new DataPoint(1, nb1),
            new DataPoint(2, nb2),
            new DataPoint(3, nb3),
            new DataPoint(4, nb4),
            new DataPoint(5, nb5),
            new DataPoint(6, nb6),
    };
    return (dp);
```

Figura 75: Código de cómo poner puntos.

Mientras que esta sería la forma de llamar a la gráfica y añadirle los valores. En este ejemplo se muestra como se cogen los valores nb0-nb6 de otra variable (n0-n6), que serían los datos que ha introducido el usuario previamente. De esta manera se obtendría la visualización de los puntos de la gráfica.

Como puede ser un poco pequeño el espacio reservado para la gráfica, se ha habilitado la opción de “Ampliar Gráfica”, la cual al ser seleccionada pasará a una nueva actividad donde sólo se encontrará la gráfica y la opción de guardarla como imagen en la galería del móvil.

A parte de apaisar la pantalla, permitiendo así un mejor aprovechamiento del espacio disponible de la pantalla; se activa la opción de agrandar la gráfica a conveniencia del usuario, pudiendo así obtener la parte que desee en mejor calidad. La opción de “Guardar” permitirá hacer una captura de pantalla a la actividad en curso y guardarla en la galería del aparato móvil, de esta manera se podrá coger como archivo JPG para su posterior utilización. Al coger

una captura de pantalla se podrá discriminar la parte más conveniente gracias a la opción ya menciona de ampliar la gráfica.

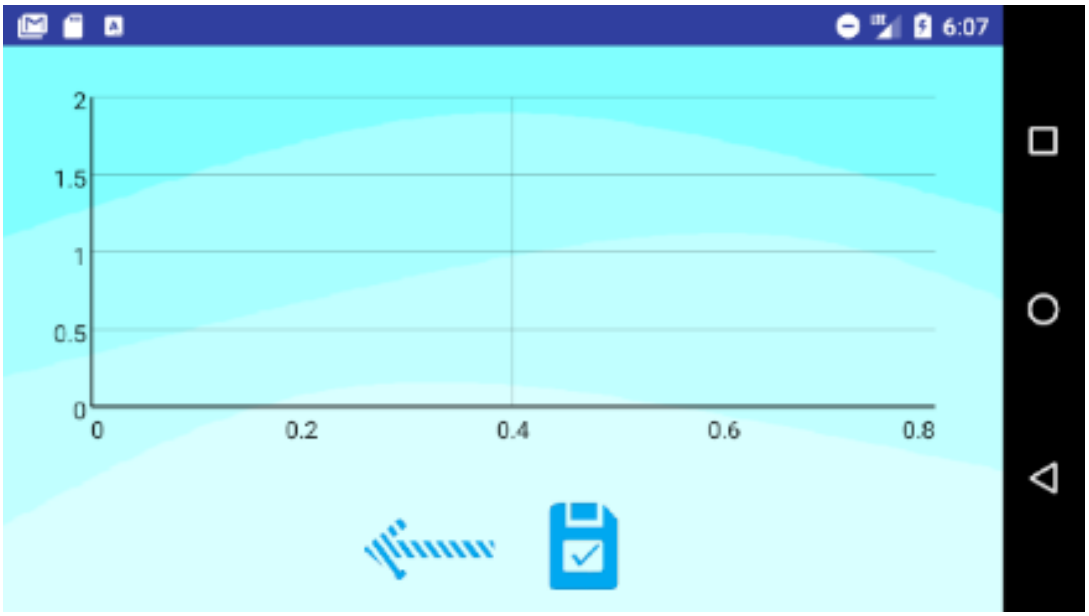

Figura 76: Pantalla de "Ampliar Gráfica".

El código que respalda la obtención de esta función es bastante tedioso y complicado, requiere de un gran número de comandos y permisos, haciendo necesarios unos conocimientos más elevados de programación que para el resto de funciones previamente escritas en el código de la app. Partiendo de la base que hay que coger una opción del teléfono, en este caso *captura de pantalla;* y modificarla para que funcione al pulsar el botón, a la vez que esa misma imagen se descarga en el teléfono en el apartado de *Galería de imágenes*.

```
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyymmsshhmmss");
String date = simpleDateFormat.format(new Date());
String name = "Img" + date + ".jpg";
String file_name = file.getAbsolutePath() + "/" + name;
File new_file = new File(file_name);
try {
    fileOutputStream = new FileOutputStream(new_file);
    Bitmap bitmap = viewToBitmap(imageView, imageView.getWidth(), imageView.getHeight());
    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fileOutputStream);
    Toast.makeText(this, "Save image success", Toast.LENGTH_SHORT).show();
    fileOutputStream.flush();
    fileOutputStream.close();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

Figura 77: Parte del código para guardar imágenes.

De todas las funciones programadas, sin duda alguna; esta es la más trabajada. El número de pasos que se tienen que seguir para que el móvil reconozca el comando al pulsar el botón, y que acto seguido guarde esa misma imagen como archivo JPG en una carpeta especifica del almacenamiento de memoria interno son muy estrictos, no pudiendo alterar ninguna línea ni cambiar el orden.

Siguiendo las pautas y consideraciones de desarrolladores más duchos en el traspaso de información en distintos formatos de una aplicación al móvil, se tiene en cuenta la posibilidad de que esta función no esté implementada del todo y pueda no funcionar correctamente en todos los dispositivos, ya que el mero cambio de los permisos requeridos o de la ubicación de la carpeta de destino haga que el programa no sea capaz de encontrar la ruta para guardar la imagen.

Dando por finalizado el grueso de la aplicación, ya que se le ha permitido al usuario introducir suficientes datos, con una gran variedad de posibilidades; y obtener los resultados más relevantes concernientes a los volantes de inercia, se puede concluir la función principal de la app. No obstante, al llegar a la pantalla de resultados final, se le dará la oportunidad de *"Dimensionar el volante"* con los datos obtenidos anteriormente.

La opción dimensionar, simbolizada con la imagen descrita en el apartado de cómo usar Solid Edge, llevará al usuario a una pantalla donde podrá meter el diámetro o el espesor y la densidad del material del que está hecho el volante para calcular el otro parámetro (espesor/diámetro) y la masa del mismo.

En la misma pantalla donde se pedirán los datos se mostrará el resultado final, no teniendo así que añadir más actividades ni códigos innecesarios.

Figura 78: Pantalla para dimensionar el volante.

La aplicación ya tiene en su base de datos algunas densidades correspondientes a los materiales más usados para la fabricación de los volantes de inercia. No obstante, si el usuario quisiese meter otra densidad distinta, se le facilita la opción de meter el valor en el recuadro debajo de las opciones al seleccionar el botón "Otro".

Para el cálculo del diámetro o espesor y de la masa del volante de inercia, se usarán las siguientes fórmulas con los datos obtenidos previamente [12][17]:

Diámetro (D)

Para la obtención del diámetro el usuario tendrá que dar el espesor **e** y la densidad $\boldsymbol{\rho}$, la inercia del volante $\boldsymbol{I_v}$ ya está calculada de antemano.

$$R = \sqrt[4]{\frac{2\,I_v}{\rho\,e\,\pi}}$$

$$D = 2\,R$$

Espesor (e)

Para la obtención del espesor el usuario tendrá que dar el diámetro **D** y la densidad $\boldsymbol{\rho}$, la inercia del volante $\boldsymbol{I_v}$ ya está calculada de antemano.

$$e = \frac{2\,I_v}{\rho\,\pi\,R^4}$$

Masa (m)

$$m = \pi\,\rho\,e\,R^2$$

Para poder ver más en detalle el funcionamiento de la aplicación y de las pantallas omitidas a la hora de explicar cómo se ha realizado el proyecto, en el apartado de anexos de incluye el *Manual de uso* donde se explicará con un ejemplo el modo en el que se debe proceder. Además del manual se detallará esquemáticamente todas las pantallas en el orden de aparición.

6. PRESUPUESTO Y ENTORNO SOCIOECONÓMICO

En el mundo de la telecomunicación, más específicamente en el de los teléfonos móvil y las aplicaciones, se mueve una ingesta cantidad de dinero, como ya se ha podido demostrar con los datos arrojados a lo largo de la disertación de este proyecto.

Refrescando los datos más significativos para el 2017 encontramos:

- Una cifra de usuarios de móviles con sistema operativo Android estimados para el 2017 de unos 4.000 millones.
- Alrededor de 270.000 millones de descarga de aplicaciones a nivel mundial.
- Un beneficio de 150.000 millones de dólares.

Todos estos datos de estimaciones, son muy alentadores y una gran motivación para meterse en este mercado dado su alto número de consumidores, el cual no parará de crecer. Pese a ser una gran oportunidad hay que ver cuál sería el verdadero público al que se dirigiría la app, ya que las predicciones son a grandes rasgos y contando todos los tipos de aplicaciones.

El público al que va dedicado es a todo el mundo, ya que no se va a hacer distinción; no obstante, el usuario tendrá que tener unos conocimientos mínimos sobre la materia de volantes de inercia, reduciendo así muy considerablemente la porción de mercado a la que se podrá dirigir la app de este Trabajo de Fin de Carrera. Esto ya se sabía, puesto que el tema a desarrollar es de ingeniería, dado que la carrera es ingeniería mecánica; por lo que no pilla de sorpresa.

Intentando estimar el número de usuarios potenciales para la aplicación, se podría pensar en el número de alumnos de ingeniería que hay en el mundo, ya que en todas las ingenierías no se dan volantes de inercia, reducir a carreras de industriales y sus derivados. Se puede suponer bien que está al alcance de todo el mundo por la incorporación del inglés a las lenguas disponibles para la aplicación.

Es muy difícil estimar el número de universidades en el mundo, ya que no hay ningún registro que englobe y cuente con todas, no obstante; se pueden encontrar cifras estimadas de entre unas 40.000 universidades. De ellas no todas imparten ingenierías por lo que se podría reducir a un 70% de esas 40.000. Partiendo ahora de 28.000 universidades de ingeniería, un 40% podría ser relacionado de industriales o similares que en algún momento den volantes de inercia.

Si se estima que cada universidad ronda los 20.000 alumnos, dado que unas universidades tienen muchas plazas y otras no tantas; se podría llegar a la cifra de 224.000.000 universitarios. Como no es una cifra verificada, se le añadirá un coeficiente del 80% para rebajar error en caso de que se desviase mucho. Llegando finalmente a unos 179.200.000 universitarios.

Un aliciente que tendrá para atraer más mercado es el hecho de que será gratuito, en sus inicios ponerlo gratis facilitará en gran medida que aumenten las descargas, y en caso de que tuviese un auge mucho mayor del que se podría esperar o alguien/alguna empresa quisiera comprarlo se empezaría a hablar del precio al que venderlo.

De momento la finalidad de este proyecto es de crear una aplicación libre de cargos para que ayude a los usuarios en los cálculos de los volantes de inercia, contribuyendo así a un mejor entendimiento de los mismos.

No obstante, es imprescindible no olvidar que todo tiene un coste y que el tiempo y recursos invertidos en la aplicación valen dinero. Por ello, aquí se desgranara todos los recursos utilizados y el coste que han conllevado.

- Formación y búsqueda de información: el tiempo requerido para usar las aplicaciones descritas (Android Studio, Snagit Editor) al nivel necesario como para hacer una aplicación de alto standing. Se incluyen aquí el tiempo de búsqueda de información requerida para manejar los programas al igual que los prototipos de ensayo que se realizaron.
- Creación de la app “Diseño de volantes de inercia” y posterior redacción del trabajo: coste debido al tiempo de desarrollo y perfeccionamiento de la aplicación final más el del escrito del informe.
- Recursos informáticos: programas utilizados y aparatos electrónicos.
- Costes indirectos: engloba el resto de costes como transporte, impresión de documentos, suministro de electricidad, material de oficina, u otros costes más difíciles de justificar.

Formación	Horas	Coste/horas	Coste final
Android Studio	20	10	200
Calculadora de prueba	30	15	450
Snagit Editor	10	10	100
Total de Formación			750

Tabla 7: Coste de formación.

App e Informe	Horas	Coste/horas	Coste final
App Diseño de volantes de inercia	220	25	5500
Informe	50	20	1000
Total de App e Informe			6500

Tabla 8: Coste de creación de la app y del escrito.

Recursos informáticos	Coste	Amortización	Coste final
Android Studio	0	-	0
Snagit Editor	0	-	0
Microsoft Office 2010	100	1/9	11.11
Ordenador	2000	1/9	222.22
Teléfono	280	1/6	46.67
Total de recursos informáticos			280

Tabla 9: Costes de recursos informáticos.

Total de costes	Coste final
Formación	750
App e informe	6500
Recursos informáticos	280
Total	7530

Tabla 10: Costes del proyecto.

A estos costes totales se le añadirán los “costes indirectos” en forma de un 10% del coste total del proyecto, ya que es una estimación bastante razonable dado la envergadura del mismo.

Total de costes	Coste final
Total del proyecto	7530
10% del proyecto	753
Total de App e Informe	8283

Tabla 11: Costes totales del proyecto.

Esta cifra total de 8283 euros refleja el sumatorio de los costes de todas las partes del proyecto, no obstante; en ningún sitio se ha aludido al importe correspondiente que habría que sumar por el IVA. En el caso de los recursos informáticos como el ordenador y el móvil, ambos costes sí que son con IVA ya que la cantidad puesta refiere al precio de compra de un establecimiento de cara al público.

7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 Conclusiones

Finalizadas las anteriores partes del proyecto, yendo desde la redacción de este mismo documento hasta la implementación y puesta en marcha de la aplicación; se puede afirmar categóricamente que los objetivos planteados al inicio del Trabajo de Fin de Carrera han sido satisfactoriamente alcanzados: se ha desarrollado una *"Aplicación informática sobre Android para el diseño de volantes de inercia."*

Siguiendo la metodología fijada y llevando un riguroso cumplimiento de la misma, nos aseguramos de tener una idea única e innovadora, nunca antes se ha llevado a cabo una aplicación de esta envergadura para el tema ingenieril de volantes de inercia a día de hoy. Además, se ha velado por la seguridad y tranquilidad de los usuarios, al haber cumplimentado estrictamente la legislación aplicable a las app; asegurando de que no se usará ninguna información privada ni se impedirá o guardará registro de la misma en ningún sitio.

La alta operatividad, la intuitiva y fácil utilización de la aplicación, la lograda estética llamativa y la elevada funcionalidad han sido el fruto de la utilización en conjunto de los programas seleccionados meticulosamente para este proyecto. La versatilidad brindada de Solid Edge para la creación de los iconos junto con Snagit Editor para el retoque y puesta a punto de las imágenes han dotado de una apariencia agradable y de fácil visualización, haciendo que la experiencia del usuario por la aplicación sea más grata.

Figura 79: Portada de la aplicación CaVI.

Android Studio ha aportado la clave para gran parte del éxito de este proyecto, permitiendo con su amplia maniobrabilidad de programación, darnos las herramientas necesarias para crear una aplicación compleja en cálculo, lo bastante abierta para poder adaptarse a las diversas necesidades del usuario.

La opción de elegir el número de vueltas del ciclo junto con el número de divisiones a la hora de representar la gráfica del par mediante la función GraphView de Android, hacen que el diseño del volante se acomode a casi cualquier requerimiento. Aportando además, una alta gama de parámetros significativos como el *"Factor de Inercia (PD^2)"*, las "*Velocidades angulares extremas (ω_{max} ω_{min})"* o la variación de energía del ciclo que es el *"Trabajo"* de nuestra máquina, factores clave para la identificación de los volantes de inercia.

También incluye la opción de guardado de datos para una posterior reutilización por parte del usuario, en caso de que quiera repetir y modificar algún parámetro; al igual que la implementación de la función para descargar la gráfica por puntos, permitiendo así llevar parte de los resultados a otro entorno de trabajo.

Este nuevo reto ha requerido de unos medios un tanto alejados de los que se podrían esperar para un ingeniero mecánico, dado que el entorno de la programación no es tan allegado para él. No obstante, esto ha supuesto que, con los medios disponibles; se enfoque de una manera distinta pero no por ello menos acertada. Teniendo en mente el objetivo de resolver problemas de volantes de inercia, y aplicando los conocimientos obtenidos a lo largo de la carrera, se hace uso de las herramientas informáticas para dar forma y soportar la idea en una aplicación para Android.

La compenetración resultante de juntar la mecánica de volantes con el desarrollo e implementación de aplicaciones para aparatos multiplataforma, hace que ambas partes cobren más valor y se complementen, *"la unión hace la fuerza"*. Pese a saber que se pueden hacer mejoras en todos los aspectos, el producto final es único y se diferencia en calidad de todo lo que hay a día de hoy en aplicaciones para móviles de cálculo de volantes de inercia.

7.2 Futuras mejoras

Siendo esta la versión más actualizada del proyecto, aún puede ser mejorada en muchos aspectos como se venía diciendo anteriormente. El tiempo que se tenía para el desarrollo era limitado al igual que la cantidad de recursos, por ello es normal y entendible que se pueda perfeccionar aún más.

Durante el proceso de codificación, se encontraron problemas a la hora de implementar ciertas funciones dadas su complejidad, algunas requerían de amplios conocimientos en el tema y dado el limitado tiempo que se tenía no se pudo profundizar en gran medida. Tener que desarrollar en código abierto aporta muchas ventajas, entre ellas la

libertad en cuanto a posibilidades; pero tiene también trae la contraparte de requerir de unos conocimientos sobre la materia ya más avanzados.

De entre todas las cosas que se podrían mejorar, las más remarcables serían las siguientes:

- Implementación y compactación del código para una más rápida compilación y ejecución. Pese a usa en su gran mayoría acciones simples, estas podrían ser reducidas mediante el uso de ciclos y funciones agrupadoras.
- Mejora de la función de guardado. Como ya se comentó en el desarrollo de la aplicación, esta función fue la más compleja y en la que más tiempo se invirtió. No obstante, se podría mejorar aumentando la precisión de la captura y la forma de guardado.
- Opciones de compartir archivos. Esta opción no se pudo poner en la aplicación por motivos de código, requería de una gran cantidad de funciones que, en un principio; hacían mal funcionar la aplicación incurriendo en el fallo de otras funciones. En próximas versiones de la app, se implementará esta opción.
- Visualización de la gráfica. La opción de usar gráficas vino de la ampliación de la librería por parte de “Graph View”, quien daba una gran cantidad de opciones para su desarrollo. Sin embargo, no tiene todas las opciones accesibles que uno quisiera, el sombreado del área bajo la curva no está del todo desarrollado, pudiendo solo sombrear de la curva hasta la parte más baja.
- Corrección de datos y comprobación de los mismos. Teniendo ciertas medidas de seguridad para que el programa no se cierre por algún dato mal metido, no es suficiente para una seguridad del 100%. Un aumento de las comprobaciones y acotaciones de los mismos haría que los resultados fuesen más realistas en caso de meter un dato alejado de lo normal.

Otras implementaciones posibles, ya menos referentes al código y funciones serían:

- Mejora del diseño gráfico y la visualización de la aplicación.
- Mejor adaptación a todo tipo de pantallas y aparatos.
- Implementación en otro tipo de sistemas (IOS, Windows, BlackBerry).
- Subida a AppStore y otras plataformas para aplicaciones.

Si se implementasen todas las mejoras descritas en la aplicación, se podría convertir en una herramienta tan avanzada o incluso más que otras del sector de hoy en día. Toda la funcionalidad y versatilidad aportada haría que su uso facilitase en gran medida los cálculos de volantes de inercia.

8. Referencias

8.1 Referencias bibliográficas

1. V.M. Faires. “Diseño de Elementos de Máquinas”. Editorial Montaner y Simón, S.A. Barcelona. 4ª Edición. 2014
2. Richard G. Budynas, J.K. Nisbett. “Diseño en Ingeniería Mecánica de Shigley”. Editorial McGraw-Hill. Octava edición.2008. ISBN-10: 970-10-6404-6
3. Cardona, S. y Clos, D. (2001) “Teoría de Máquinas” (Edición en Castellano). Ed.: UPC. Barcelona.
4. Norton, R.L. “Design of machinery: an introduction to the synthesis and analysis of mechanisms and machines”. 5th ed. McGraw Hill, New York, 2012
5. Shigley, J.E. & Uicker, J.J. “Teoría de máquinas y mecanismos”. McGraw Hill, 1998, ISBN 968·451·297·X.
6. S. Hébuterne, S. Pérochon, “ANDROID guía de desarrollo de aplicaciones para Smartphones y tabletas”. Eni ediciones, segunda edición, 2014.
7. Neil Smyth, “Android Studio 2 Development Essentials”, CreateSpace Publishing, 2014.
8. Lluís Ripoll Masferrer, Tesis doctoral, “ANÁLISIS Y DISEÑO DE VOLANTES DE INERCIA DE MATERIALES COMPUESTOS”, 2005, Universitat Politècnica de Catalunya
9. Maria Inés Lopes Marques. Trabajo de fin de master “Design and Control of an Electrical Machine for Flywheel EnergyStorage System”. 2008, Instituto Superior Técnico, Universidade Técnica de Lisboa.
10. Anirudh Pochiraju, Thesis for the degree of Master of Science, “Ddesign principles of a flywheel regenerative braking system (f-rbs) for formula sae type racecar and system testing on a virtual test rig modeled on MSC Adams”, University Of Kansas School Of Engineering. 2012
11. Satish Samineni. Thesis for the degree of Master of Science. “Modelling and Analysis of a Flywheel Energy Storage System for Voltage Sag Correction”, 2003, University of Idaho.
12. José Ramón Ureña Marín. Tesis doctoral “Estudio histórico-tecnologico de la producción de azúcar de caña: aplicación al análisis desde la ingeniería industrial y la ingeniería gráfica de máquinas de vapor Fives-Lille y Mirrless-Watson en la costa granadina”, 2012, Universidad de Jaén. (5.5.3.1 Calculo del peso del volante).
13. A. de Lamadrid, A. de Corral. “Cinemática y Dinámica de Máquinas”, Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingeniería Industrial, Sección de Publicaciones, 1969.
14. A. Rodríguez de Torres, J. Martel, P.R. Moliner, “Elementos de Máquinas”, Publicado por la UNED, 1976.
15. Ruth Sáez Blázquez, Yaiza Sanz Ugena, Carlos Serrano Hernández, Alejandro Tejero Castellanos, Yago Uzquiano Díez, Trabajo de la asignatura “Cinemática y dinámica de máquinas” UC3M, “Cálculo de volantes de inercia (YARCY3.2)”, 2014.

16. Miguel Esteban Lineros, Trabajo de Fin de Grado “Desarrollo de una aplicación informática sobre Android del diseño de un dispositivo mecánico”, 2016, Universidad Carlos III de Madrid.
17. Apuntes de la asignatura “Teoría de máquinas”. (Curso 2014-2015) Departamento de Ingeniería Mecánica. Universidad Carlos III de Madrid

8.2 Referencias web

18. Las aplicaciones, el nuevo gran filón de la industria de móviles. Último acceso (8/05/2017)
http://www.expansion.com/2015/02/27/empresas/tecnologia/1425069142.html

19. La venta de 'tablets' se hunde en 2016: 55 millones de unidades menos. Último acceso (8/05/2017)
http://www.lavanguardia.com/tecnologia/20170203/413978333603/tablets-ventas-samsung-galaxy-tab-amazon-kindle-huawei-tab-windows-10-ipad-ipad-pro.html

20. Los smartphones Android siguen aplastando en ventas a los iPhone. Último acceso (8/05/2017)
https://ando4all.com/2016/01/smartphones-android-aplasta-en-ventas-iphone-ios-couta-mercado

21. Las sorprendentes cifras que mueve el mundo del móvil. Último acceso (8/05/2017)
http://www.media-tics.com/noticia/7301/moviles-/las-sorprendentes-cifras-que-mueve-el-mundo-del-movil.html

22. Android Studio. Último acceso (8/06/2017)
https://developer.android.com/studio/index.html?hl=es-419

23. Graph View.Último acceso (8/06/2017)
http://www.android-graphview.org/

24. Tutoriales de diseño de aplicaciones por PRABEESH R K. Último acceso (8/06/2017)
https://www.youtube.com/user/TICOONTECHNOLOGIES/videos

25. Ramon invarato Mendez, "Android 100%", 2014. Último acceso (8/06/2017).
https://jarroba.com/libro-android-100-gratis/

26. Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. Último acceso (5/06/2017).
http://www.gartner.com/newsroom/id/3609817

27. Historia de Android: La Evolución a lo largo de sus versiones .Último acceso (12/05/2017)
http://androidzone.org/2013/05/historia-de-android-evolucion-versiones/

28. ECLIPSE. Último acceso (13/04/2017).
https://eclipse.org/

29. Basic4Android: Desarrollando aplicaciones para Android al estilo Visual Basic. Último acceso (12/05/2017).
http://developeando.net/basic4android/

30. Aplicaciones móviles y Ley de Comercio Electrónico (LSSI-CE). Último acceso (25/05/2017).
https://conversiaconsultinggroup.wordpress.com/2014/08/11/lssi-aplicaciones-moviles/

31. Guía de Protección de Datos para desarrolladores de aplicaciones móviles. Último acceso (12/05/2017). https://ayudaleyprotecciondatos.es/2016/06/06/normativa-lopd-aplicaciones-moviles/

ANEXOS

ANEXO I: PANTALLAS DE LA APLICACIÓN CaVI

En el siguiente anexo se agrupan todas las pantallas que el usuario va a encontrar en la aplicación CaVI, todas las pantallas se han sacado de la versión en Español ya que las de la versión en Inglés son una mera traducción.

Empezando por la portada de la aplicación donde el usuario podrá seleccionar el idioma de la app.

Figura 80: Portada de la aplicación CaVI

Tras elegir el idioma, en este caso Español; se mostraría la pantalla del menú, donde se encontrarían las opciones de teoría, manual de uso, calcular el volante y símbolos.

Figura 81: Menú

La opción de símbolos viene a explicar al usuario el significado de cada uno de los botones de la aplicación, ya que la gran mayoría están representados por una imagen. El botón para la teoría redirecciona al usuario a una pantalla deslizante donde aparecen los fundamentos teóricos sobre los volantes de inercia. El manual de uso da la posibilidad de visualizar dos ejemplos de cómo se usa la aplicación. Y la opción de calcular volante abrirá la pantalla para ir seleccionando el tipo de problemas a resolver.

Figura 82: Símbolos

Todos los botones se han agrupado y puesto su significado para que el usuario, en caso de duda pueda asegurarse qué comando acciona cada función. Al no aparecer las letras en cada pantalla, tenerlas todas agrupadas en una sola imagen hace que sea fácil y de rápido remedio saber cuál es cual.

El manual de uso permite al usuario elegir entre dos ejemplos, uno de par motor constante y otro de par resistente constante; abarcando así las dos opciones que se dan a la hora de calcular los volantes.

Figura 83: Manual de uso

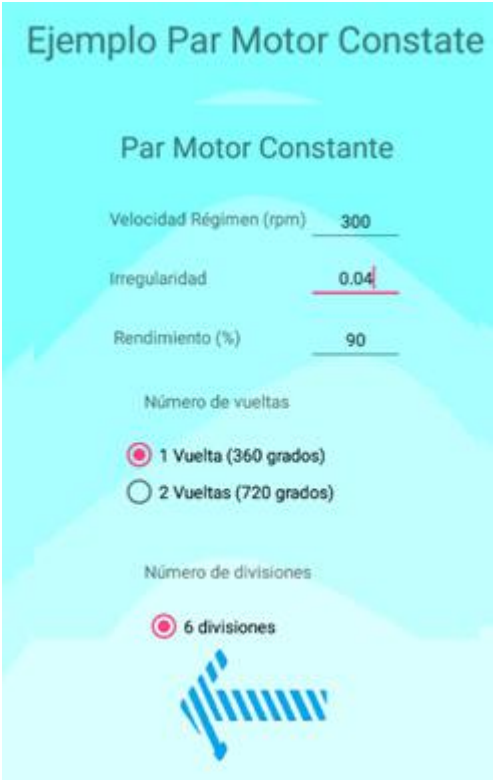

Figura 85: Ejemplo Par Motor Constante

Figura 86: Ejemplo Par Resistente Constante

En la pantalla de teoría, un recuadro scroll contendrá toda la información relativa a los volantes de inercia, las fórmulas significativas y la manera de usarlas. El usuario verá el procedimiento que se ha utilizado para obtener los valores que en la pantalla de resultados aparecen.

Figura 87: Teoría

Figura 88: Cálculo de Volante

Seleccionando el tipo de ejercicio a resolver, eligiendo entre par motor constante o par resistente constante; el usuario procederá a introducir los datos correspondientes.

Figura 89: Menú par resistente

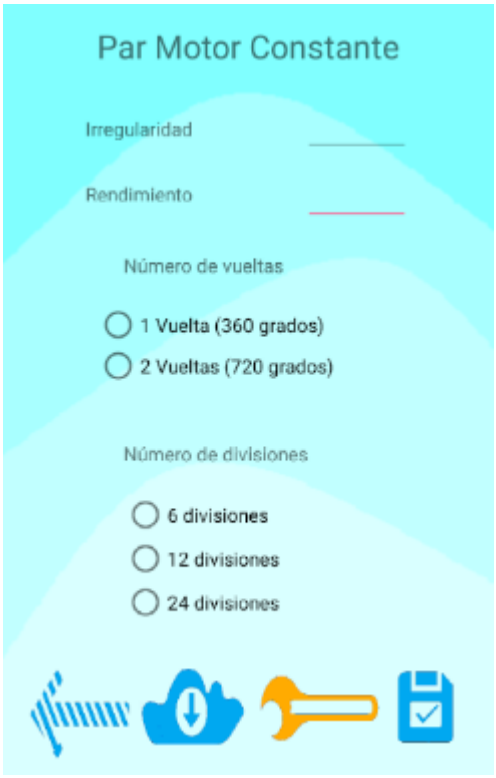

Figura 90: Menú par motor

Tras introducir los datos de velocidad de régimen, irregularidad y rendimiento; se seleccionaran el número de vueltas y divisiones de la gráfica. Acto seguido el usuario podrá rellenar las casillas con los respectivos valores de los puntos de la gráfica de solicitaciones.

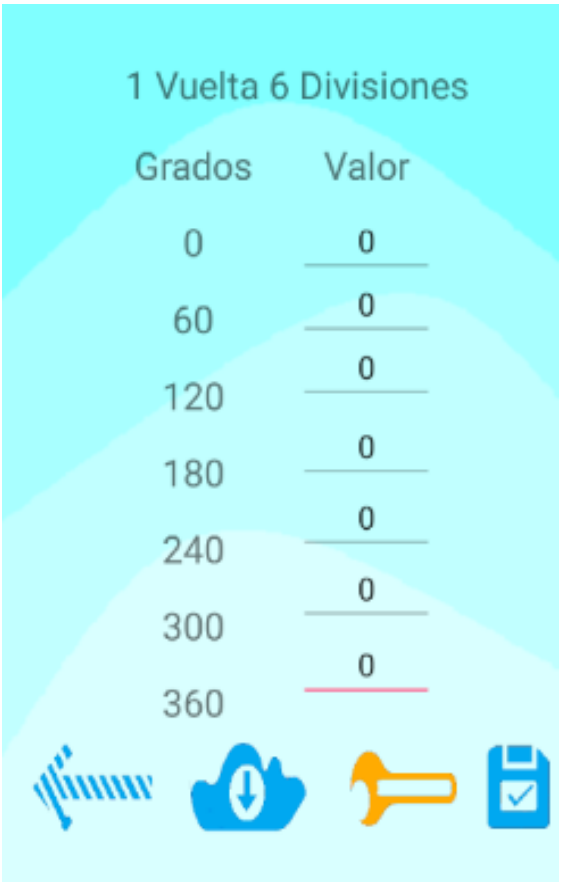

Figura 91: 1 Vuelta 6 Divisiones

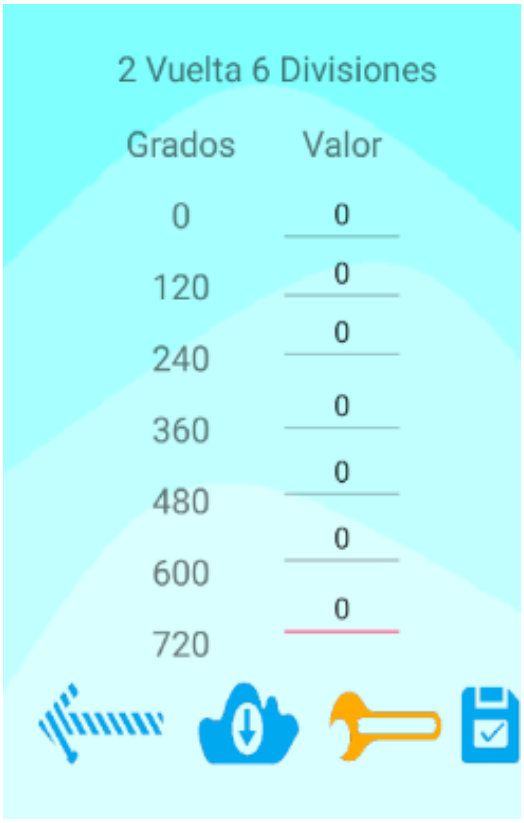

Figura 93: 2 Vuelta 6 Divisiones

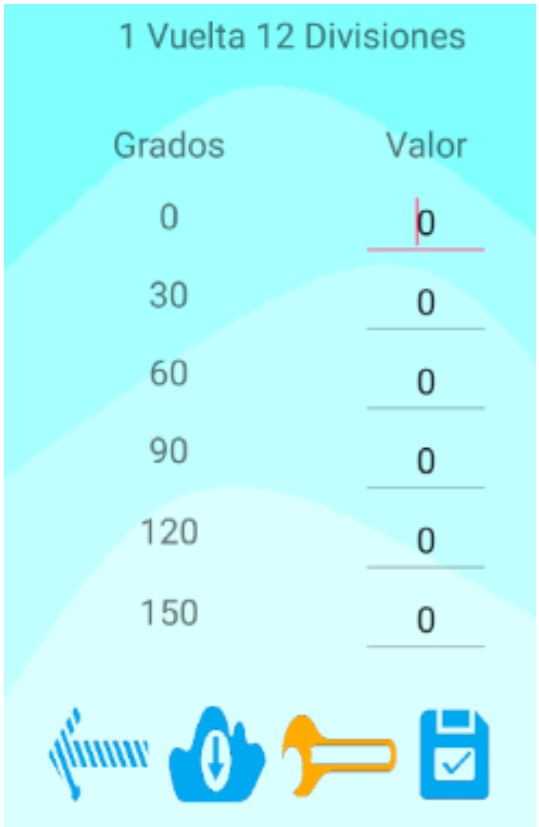

Figura 92: 2 Vuelta 12 Divisiones

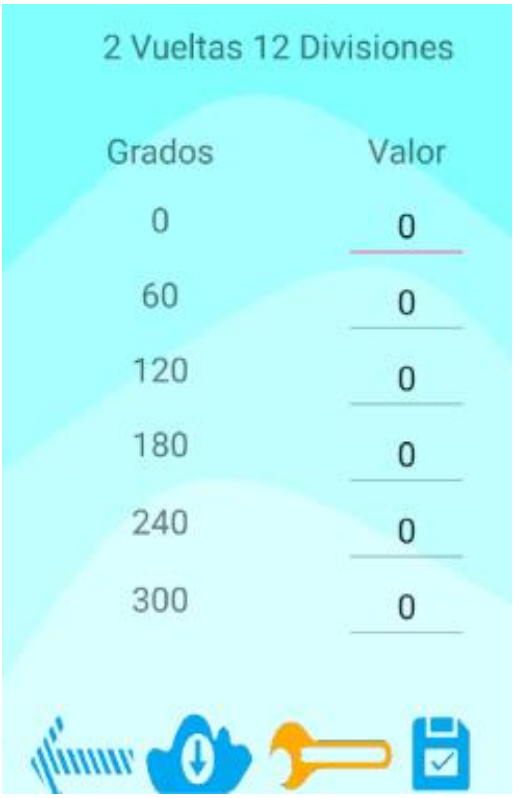

Figura 95: 2 Vueltas 12 Divisiones

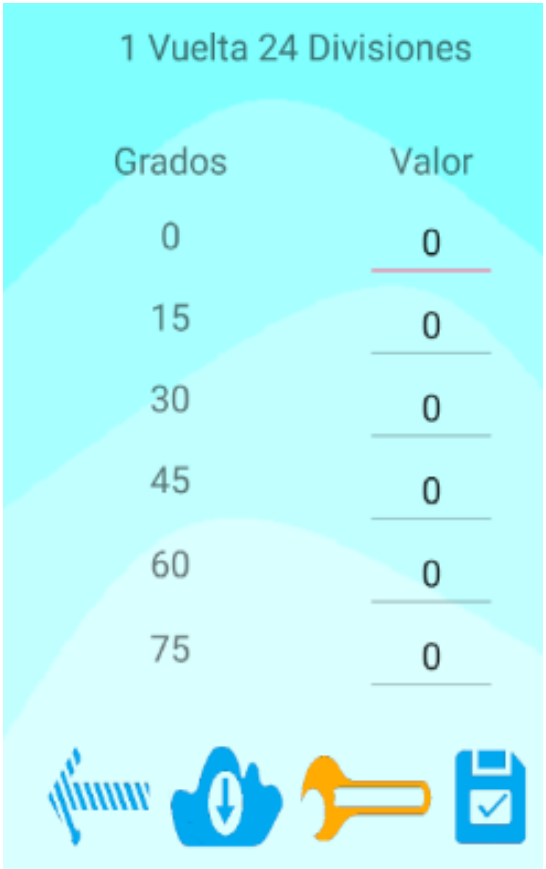

Figura 94: 1 Vueltas 24 Divisiones

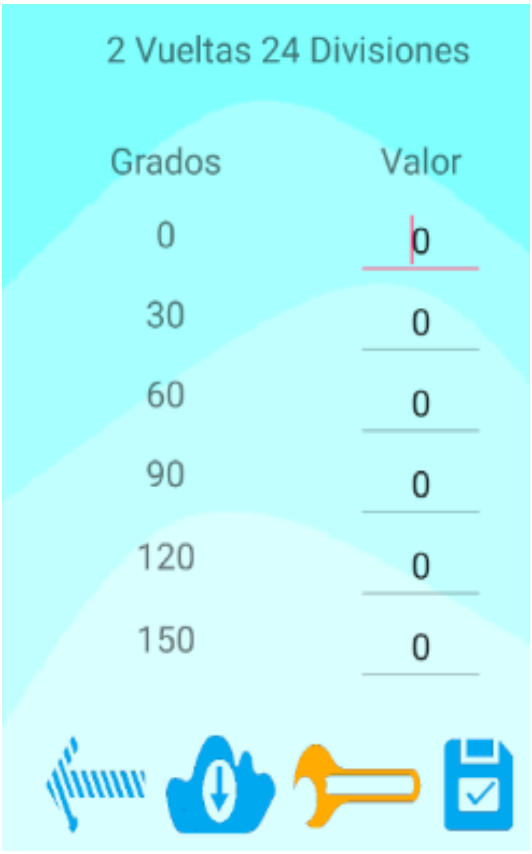

Figura 96: 2 Vueltas 24 Divisiones

Una vez rellenados los datos de los puntos seleccionados, se procederá a la resolución del problema y a la visualización de los parámetros clave junto con la gráfica creada.

Figura 97: Solución

Para poder ver mejor la gráfica creada se pulsaría el botón *"Ampliar Gráfica"* el cual haría que se pasase a una pantalla con solo la gráfica en grande y apaisada.

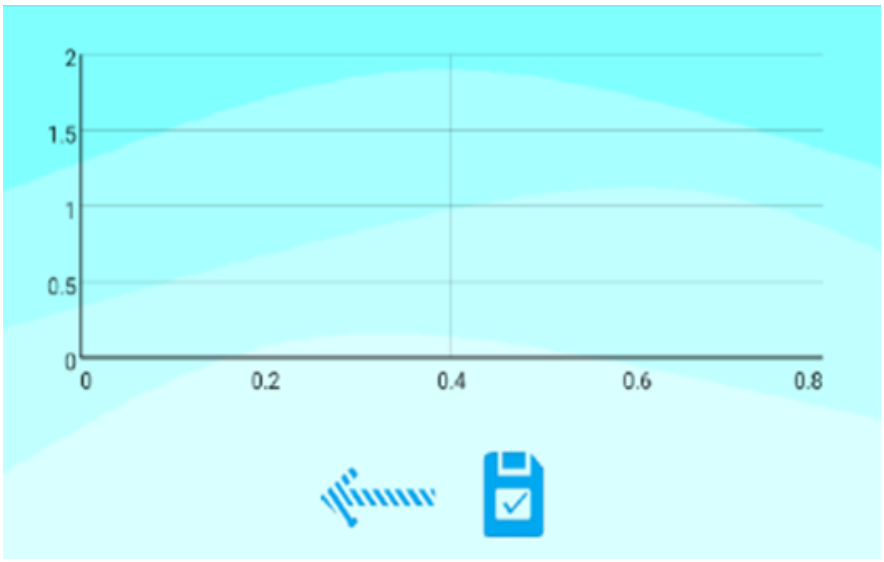

Figura 98: Ampliar Gráfica

Por último se puede dar medidas al volante de inercia calculado mediante la opción *"Dimensionar"*.

Figura 99: Dimensionar

ANEXO II: MANUAL DE USO

En la siguiente colección de imágenes, se presentará el contenido de uno de los ejemplos del manual de uso, más específicamente, el ejemplo de "Par Motor Constante". Los botones seleccionados se marcarán en naranja para facilitar el seguimiento de las operaciones.

Comenzando por la selección de la opción *"Manual de Uso",* nos aparecerá la siguiente opción de elegir entre *"Ejemplo Par Motor Constante"* y *"Ejemplo Par Resistente Constante"*, de entre ambas seguiremos con el ejemplo de par motor.

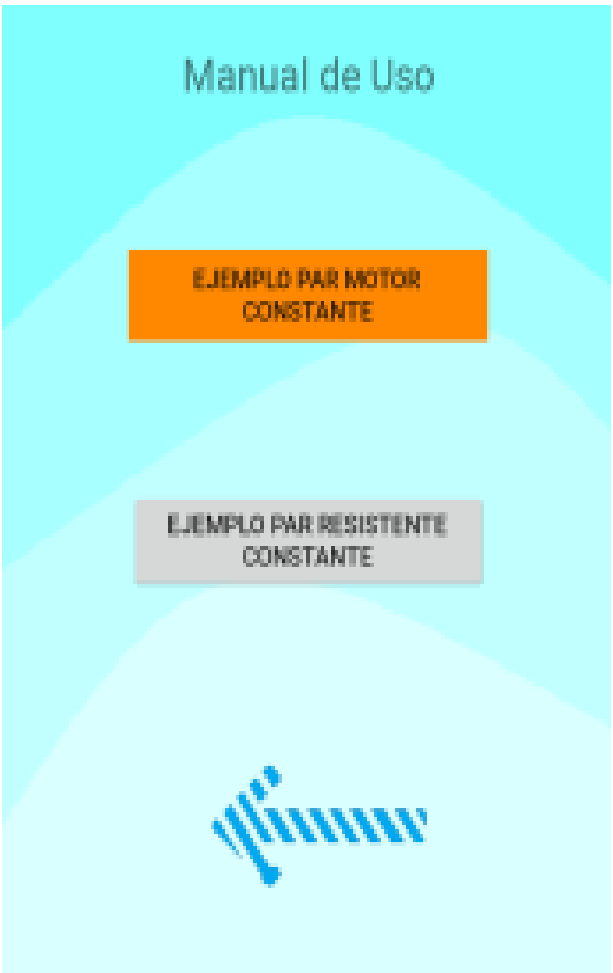

Figura 100: Manual de uso, opción Motor Cte

Tras elegir esa opción aparecerá el menú del par motor, en él elegimos las siguientes opciones:

Velocidad de régimen (rpm)	300
Irregularidad	0.04
Rendimiento (%)	90

Tabla 12: Datos

Una vuelta seis divisiones.

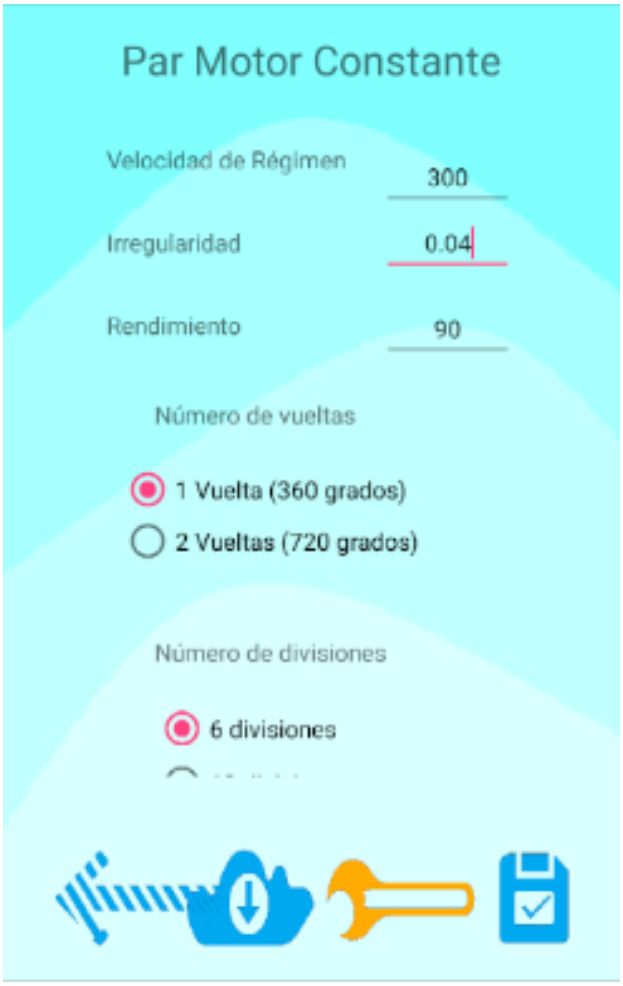

Figura 101: Par motor Constante

Metiendo uno a uno los datos de los puntos, se obtendría:

Grados	Valor
0	100
60	100
120	150
180	50
240	50
300	-100
360	0

Tabla 13: Valores de la gráfica

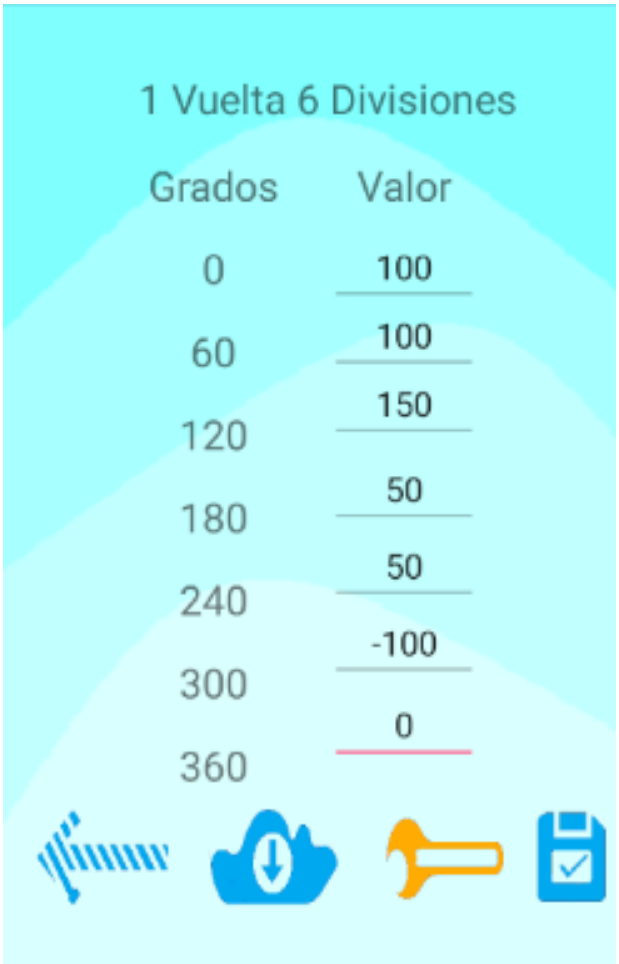

Figura 102: Valores de la gráfica

Ya daos todos los valores necesarios se obtiene por fin los resultados.

Figura 103: Solución ejemplo

Par Medio (N m)	50
Potencia Motor Necesario (W)	1745.32
Momento de Inercia (kg m^2)	4.64
Factor de Inercia (N m^2)	182.15
Velocidad Angular Min (rad/s)	30.78
Velocidad Angular Max (rad/s)	32.04
Trabajo (J)	183.25
Energía Cinética Min (J)	2197.48
Energía Cinética Max (J)	2382.25

Tabla 14: Resultados ejemplo

Si queremos ampliar la gráfica, con darle al botón, obtendríamos:

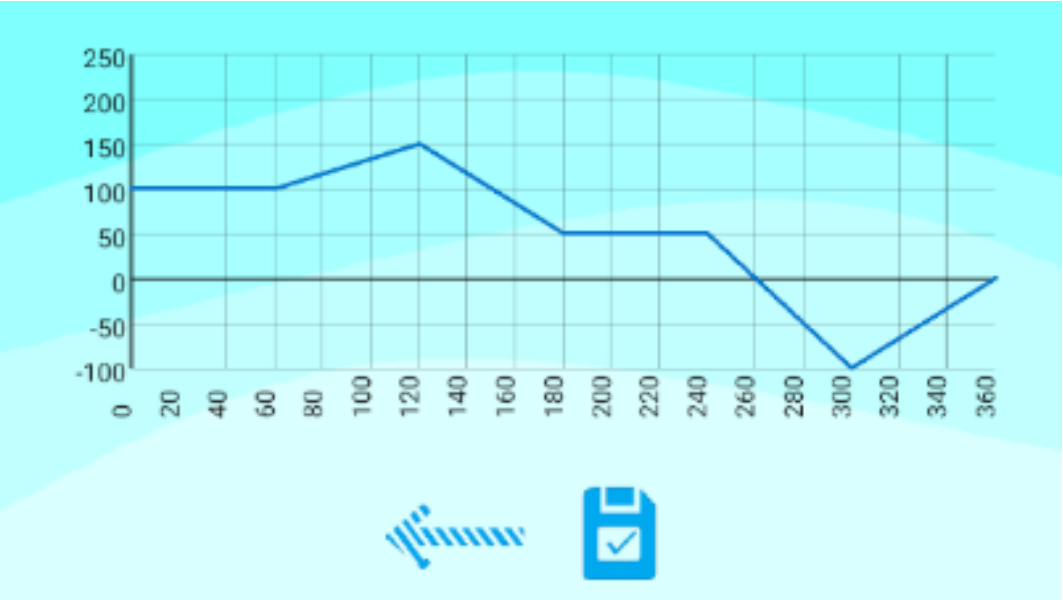

Figura 104: Gráfica ampliada